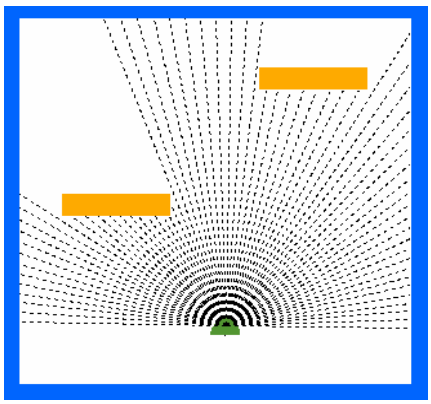
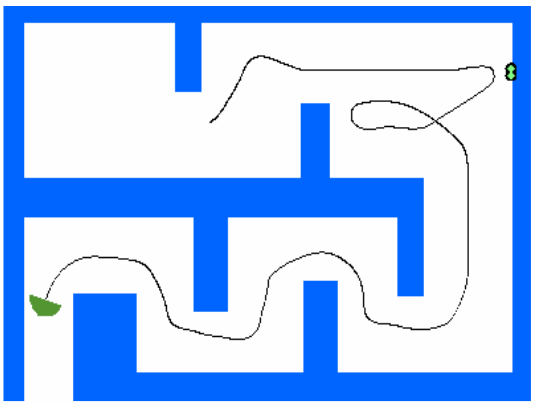
Technical report, IDE0624, January 2006

A Neural Reinforcement Learning Approach for Behaviors Acquisition in Intelligent Autonomous Systems

Master's Thesis in Computer Systems Engineering

Eric Aislan Antonelo





School of Information Science, Computer and Electrical Engineering
Halmstad University

A Neural Reinforcement Learning Approach for Behaviors Acquisition in Intelligent Autonomous Systems

Master's thesis in Computer Systems Engineering

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

January 2006

Description of cover page picture: Left, Image from the developed simulator.
Right, Image from the developed simulator.

Acknowledgments

Firstly, I would like to thank my thesis advisor, Professor Albert-Jan Baerveldt, for his essential aid on special issues during my graduate studies at Halmstad University. His enthusiasm and will to help me have been indispensable factors for the development of this work.

I am very grateful to Dr. Thorsteinn Rögnvaldsson for his important support and feedback. His challenging mind has helped me enormously towards the refinement of this work.

Numerous thanks to Dr. Mauricio Figueiredo for his fundamental insights on my research work. His odd way of thinking has taught me invaluable things. Without his unreserved motivation, my graduate studies in Sweden would not have come true.

I would like to thank Dr. Jörgen Carlsson for his indispensable aid on bureaucratic matters with the scholarship program. He is one of the people responsible for my entrance in the Master's Program in Computer Systems Engineering at Halmstad University.

Special thanks to the ALBAN Office and to all people that, in some way, have made possible my dream come true: to be granted with a ALBAN scholarship for studying in Europe (*).

Thanks to Halmstad University and, in special, to the School of Information Science, Computer and Electrical Engineering for making possible this work.

I want to thank my friends that have been always supporting me, here in Sweden, and in Brazil. They have had an important role during my studies abroad.

Finally, my warm thanks to my family that is the source of my forces. They are the only one who can provide me unique care and motivation. This work would be impossible without their existence.

(*) This work is supported by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship no. E04M027421BR.

Eric Aislan Antonelo
Advisor: Albert-Jan Baerveldt
Co-advisors: Thorsteinn Rögnvaldsson
Mauricio Figueiredo (State University of Maringá, Brazil)

Halmstad University, Sweden
January 2006

Abstract

In this work new artificial learning and innate control mechanisms are proposed for application in autonomous behavioral systems for mobile robots. An autonomous system (for mobile robots) existent in the literature is enhanced with respect to its capacity of exploring the environment and avoiding risky configurations (that lead to collisions with obstacles even after learning). The particular autonomous system is based on modular hierarchical neural networks. Initially, the autonomous system does not have any knowledge suitable for exploring the environment (and capture targets – foraging). After a period of learning, the system generates efficient obstacle avoidance and target seeking behaviors. Two particular deficiencies of the former autonomous system (tendency to generate unsuitable cyclic trajectories and ineffectiveness in risky configurations) are discussed and the new learning and control techniques (applied to the autonomous system) are verified through simulations. It is shown the effectiveness of the proposals: the autonomous system is able to detect unsuitable behaviors (cyclic trajectories) and decrease their probability of appearance in the future and the number of collisions in risky situations is significantly decreased. Experiments also consider maze environments (with targets distant from each other) and dynamic environments (with moving objects).

Contents

1 Introduction

1.1 Motivation

The exploration of unknown environments in the real world has constantly caused a great appeal not only to the scientific community, but also to the humanity in general. The implementation of mobile robots for such use is not an easy task.

Most of current mobile robots operate in simple and controlled environments. In general, these robots are assisted by placement of special landmarks in the target environment, indicating the path to be followed [1]. In this context, the development of autonomous robots, which are adaptive to complex and unknown environments, is a reason of intense research.

Usually it is possible to understand mobile robot navigation as a problem of establishing trajectories, so that tasks (goals) are accomplished with acceptable performance. The vehicle tasks could be, for example: to capture targets, to deviate from obstacles, to follow walls, to recharge batteries, to exploit the environment, etc.

Important applications come up from the development of mobile robots that are capable of operating in complex environments: surface cleaning (floors, industrial tanks, ship hulls, external structures of buildings, ducts, etc.), transport of materials, e.g., crop of agricultural products, and vigilance systems.

Research about navigation systems has been done in several ways, depending on the characteristics of the environment, the robot model, the type of the task, and the performance criteria [1 and 11]. A class of navigation systems, the autonomous systems, has captivated the scientific community not only because of the challenge involved but also because of the strategic importance.

Such systems determine the robot trajectory in an unknown environment without external help. In this context, navigation systems learn their own navigation strategy from the environmental interactions (based on their own experiences) [1].

The term autonomy is usually utilized in a broad sense. In this case, a robot that moves by itself (without external aid) is considered autonomous. Differently, here a system is considered autonomous if it is capable of generating (learning) the rules that govern its behavior and consequently, that improve its performance on the environment task. It is important to note that the learning takes place through the system (robot) interactions with the (unknown) environment.

Traditional control engineering techniques that require the use of complete environmental models for the navigation task are not well suited for complex and uncontrolled environments. This is because these environments are uncertain by nature (it presents unpredictable events) and can not be modeled once there are an infinite number of events. According to [20], these navigation systems would be called automatic (differently from autonomous), that is, their responses are based on known modeled situations that generate repetitive pre-incorporated behaviors.

Computational intelligence based strategies (e.g., neural networks and evolutionary systems) have been disseminated as viable and powerful alternative [2, 4, 5 and 7]. Neural networks confer learning features to self-guided mobile robots. In this sense, a robot becomes adaptive to its environment, for instance, it can learn new (navigation) strategies according to exposure to unknown situations.

Such systems are sometimes referred in the literature as embodied intelligent systems. In [35], Steels define such (intelligent) systems as systems that have four properties, namely: self-

maintenance, adaptivity, information preservation, and increase in complexity. He defines intelligence upon the perspective of biological systems.

In this sense, the system proposed in this work is inspired according to the ideas aforementioned: the environment is complex and unknown; the autonomous system develops its own (navigation) strategies through interaction with the environment; specific behaviors and navigation strategies are molded by navigation experiences (succeeding as the robot navigates) according to classical reinforcement learning procedure [18] and operant conditioning procedure [29]. The design of the autonomous system follows the particularities of biological systems (animal behavior [21] and neuronal group selection theory [10]). Furthermore, the architecture of the system is composed of hierarchical neural networks.

1.2 Context

The first generation of mobile robots dowered with a navigation system comes from the sixties. Shakey, a mobile robot developed by Nilsson in 1969 [22], had a navigation system that worked in an environment with special demarcated objects. Its perception module provided information for another module that used symbolic inference rules to generate navigation decisions.

Other robots were also based on symbolic processing in the first decades of research on the area: Hilare in 1977 at Laboratoire d'Automatique et d'Analyse dês Systèmes, France [23], Cart in Stanford [24], and Rover at CMU [25]. Such robots based their decisions on an internal world representation that was used for path planning, characterizing them as deliberative systems (Figure 1.1).

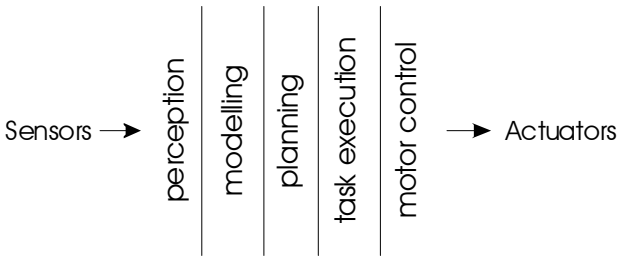


Figure 1.1: Sequential (functional) modules execution for a mobile robot control system.

There are some shortcomings with this approach. One is the cost associated with the internal world representation and the amount of information needed for constructing it. Several real situations that can take place when the system is running should be treated and coded in the knowledge base in form of rules. This can lead potentially to a great knowledge base and a great associated implementation effort.

Ronald Brooks suggested a new architecture for controlling mobile robots in 1986 [13], called Subsumption Architecture. This new model is based on task-achieving behaviors organized in different layers of control. Each class of behaviors is implemented in a specific layer (Figure 1.2).

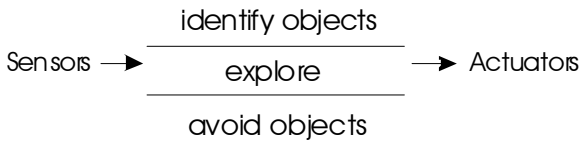


Figure 1.2: Layers of behaviors for a mobile robot control system.

Brooks defines a *level of competence* as an informal specification of a class of behaviors that the robot can present. Each such level is implemented in a layer. Depending on the current situation, a higher level of competence can suppress the output of lower levels of competence in order to attend the current robot priorities.

The system designed by Brooks aims at robustness (the system should work even when some sensors fail or when the environment changes drastically) and extensibility (more capabilities can be added to a robot). Besides, it is a behavior-based system and, unlike deliberative systems, is not based on cognitive processes on an internal world representation. Interestingly, there is no unique agreement on the difference between reactive and behavior-based systems. Mataric [26] presents distinct characteristics for both classes of systems. On the other hand, Arkin [11] considers behavior-based systems as a specific class of reactive systems.

The control system created by Brooks is adaptive to some extent. Once the system can not generate new behaviors (through a learning process) according to environment dynamics and characteristics, it lacks on adaptability.

Proposals of autonomous navigations systems that use neural networks have been increasingly developed in the literature [2 to 6]. The potential for learning associated with neural networks is a feature that confers autonomous navigation systems high adaptability to the environment.

Proposals in [4] and [5] present the behavior-based approach associated with the (classical) reinforcement learning. Two repertoires reproduce instinctive behaviors of target seeking and obstacle avoidance (basic behaviors). At first these repertories work independently, generating conflicting behaviors, e.g. if an obstacle is between the robot and the target. A third repertory is assigned for coordination of the innate behaviors. Initially the performance frustrates the expectations, since the autonomous system does not have an expert knowledge about navigation. As the navigation proceeds, environment interactions provide the bases for a reinforcement learning strategy. In the end the navigation system guides efficiently the robot to the target.

In a different way, the system proposed in [7] does not have a behavior-based architecture, that is, there are not instinctive behaviors a priori. The classic reinforcement learning is integrated to a learning classifier system to conceive an evolutionary navigation system. The evolutionary system learns simultaneously obstacle avoidance and target seeking behaviors, and it learns to coordinate them as well.

Several experiments within the perspective of embodied intelligence are presented in [27]. For instance, at the Swiss Federal Institute of Technology (EPFL), evolutionary experiments without human interaction (including competitive and cooperative co-evolution) are carried out in simulation and in real environments (using the robot *Khepera* [33]). Interesting results are presented showing that complex behaviors can be achieved through artificial evolution of a population of individuals (e.g., neural circuits, robot controllers) - that constantly interacts with the environment [27 and 34].

Such as the aforementioned references, most research reports in the literature model the autonomous navigation problems considering robots are equipped with target and obstacle

sensors, and learn to generate obstacle avoidance and target seeking behaviors. The problem model described in [28] presents, at least, two innovative and curious aspects: a) there are different classes of objects in the environment, each of which associated with a respective color and; b) the sensors, to detect the colors of the objects as well as the distance from the objects to the robot, are not specific either for targets or obstacles. The autonomous system consists of a hierarchical and modular neural network with also innovations on the characteristics of neuron units and neuron connection arrangements. Exploiting navigation experiences, the autonomous system learns, becoming able to distinguish objects (generating specific behaviors for each one of the different classes of objects). Furthermore it learns to take navigation decisions to move the robot to the closer attractive objects (targets).

1.3 Objectives and Results

The general objective of this work is to devise new artificial learning techniques and innate control mechanisms for application in autonomous behavioral systems for mobile robots.

An autonomous system existent in the literature [28] is of special interest for this work. Through executing simulations experiments, it is possible to observe deficiencies of the former system. Firstly, the performance of the mobile robot (with respect to its capacity of exploring the environment) guided by the autonomous navigation system is strongly dependent on simulation initial conditions (e.g., initial position of the robot and environment configuration). The actual result is: the mobile robot guided by that system can be easily trapped in an endless cyclic trajectory, what eliminates exploiting possibilities and further learning (navigation experiences).

Secondly, the original autonomous system is not able to stabilize the number of collisions with obstacles after a long learning period in certain environments. Risky configurations for the robot (that often leads to inevitable collisions) are the cause of this poor performance with respect to number of collisions.

An artificial learning mechanism (based on operant conditioning [29]) is devised in order to diminish the probability of appearance of cyclic trajectories in the future. This learning mechanism is fired when a monotony event is detected, that is, when there have been no contacts with objects (either obstacle or target) for a certain time interval (probably indicating a cyclic trajectory). Simulation results show that the robot (guided by the autonomous system) is not endlessly trapped in a cyclic trajectory: the system gradually learns to develop a suitable exploring behavior (and also forage) independently on simulation initial conditions.

In order to tackle the second mentioned deficiency, a new innate control repertoire is proposed to integrate the architecture of the system. This control repertoire can detect eventual risky situations and adjust appropriately specific parameters of neurons associated with concept of proximity. The parameters influence the output of neurons and the way the competition between neurons is accomplished. The adjustments are such that the sensibility of the neurons is changed in order to cope with the new requirements of a risky situation: move the robot away from the undesired configuration. Simulation results confirm the effectiveness of the innate control repertoire integrated in the architecture of the system.

Other diverse experiments are accomplished. Simulations using a simple environment (with no obstacles inside but with 2 targets) show that with very few collisions the autonomous system learns to generate behaviors suitable for avoiding repulsive objects (obstacles) and for capturing targets (that are indefinitely replaced in the environment). Generalization capabilities are also

confirmed: first the autonomous system learns in a particular environment and in sequence it is tested in a distinct environment (generating trajectories free of collisions). A maze environment (where 5 targets are distant located from each other) is also used in the simulations. In the best case (i.e., simulation experiment resulting in very good performance), the robot captures each target at least 5 times. Dynamic environments (with moving objects) are also considered in this work. From simulation results it can be seen that the autonomous system is able to adapt to the environment dynamics, generating a trajectory free of collisions with a moving obstacle in the best case. Finally, experiments with multiple robots in the same environment are accomplished: the cooperation achieved when robots work together influences positively the performance with respect to the total number of target captures.

The performance of the autonomous system is always measured with respect to the capacity of environment exploration and with respect to the number of collisions with obstacles and to the number of target captures.

2 Background

2.1 Introduction

Several problems concerning robot autonomous navigation have great complexity. Traditional techniques are considered unacceptable to be adopted in the solution of these problem types (where the environment is considered unknown, dynamic and unpredictable) because there are high costs associated with system modeling and with the amount of computational resources needed for implementation.

Computational techniques inspired in nature constitute important tools for solving complex problems whose modeling is considered a hard task. These tools present very particular characteristics once they are based on biological systems, e.g.: robustness, fault tolerance, parallel processing, adaptability, learning, etc. In this way, computational intelligence based techniques (that are inspired in nature) are considered real alternatives to traditional techniques when a problem has great complexity. Furthermore, they provide efficient results using relatively-simple models.

This work uses computational intelligence based techniques for the solution of the problem described in Section 1. Artificial neural networks (a sub-area of computational intelligence) are considered a basic substrate for the implementation of the proposed model. An introduction of this theory is given in Section 2.2 and Appendix A. Several theoretical concepts are used to support the proposed model, such as: conditioning, reinforcement learning and neuronal selection theory. They are described in Section 2.3 and Section 2.4.

2.2 Neural Networks

Research in artificial neural networks has been motivated mainly by its distinct way of information processing when compared to digital computers. Since they are inspired in the human brain, some powerful capabilities of information processing are also borrowed from the biological counterparts. A more complete introduction about neural networks is given in Appendix A.

The psychologist Donald Hebb proposed the base of associative learning (at cellular level) [15]. According to Hebb learning postulate, if two connected neurons fire repeatedly and simultaneously, the respective synapse is selectively strengthened; on the other hand, if two neurons fire repeatedly but asynchronously, the synapse between them is selectively reduced.

In this way, a neuron A is able to increase its influence on the activity of a neuron B (considering they are connected), as well as it is able to decrease its influence (depending on the synchrony of firings). The simplest form of Hebb learning is given by:

$$\Delta w_{kj}(n) = \eta y_k(n) x_j(n)$$

where, η is a positive constant that determines the learning rate, y_k is the output signal (post-synaptic activity) and x_k is the input signal (pre-synaptic activity). This equation emphasizes clearly the co-relational nature of a Hebbian synapse.

2.3 Learning and Conditioning

The phylogenesis of a species consists of a series of events elapsed along many generations that influence in the determination of this specie's characteristic features. The phylogenesis determines innate mechanisms as well as the capacity of emergence of certain behaviors through learning.

Some behavioral features are so characteristic of a species as its anatomical features [12]. The simplest forms are called reflex, innate mechanisms present in animal species and being a result from natural selection (e.g., sneeze, blink, dodder in the human's case). Furthermore, reflexes are deeply related to healthy maintenance and survival and reproduction promotion.

An unconditioned reflex is an innate and biologically established reflex. It is expelled from an unconditioned stimulus, i.e., from a stimulus that naturally incites it (e.g., a salivation reflex in the presence of food inside a mouth). A neutral stimulus is an event that does not provoke any type of reflex action.

The evolutionary mechanism of natural selection underlies the following conclusion: species fitness is increased when suitable reflexes are available in the necessary moment. This is the reason why animals hold a simple learning mechanism for associating reflexes and stimuli. Two basic mechanisms are very known, classical (respondent) and operant. Since Ivan Pavlov's work [18], it is known that an animal species can present conditioned reflexes that are characterized by the history of coupling between a neutral stimulus and an unconditioned stimulus. From successive associations between these two stimuli, the previously neutral stimulus will be able to incite the reflex that was only provoked by the unconditioned stimulus. This basic learning mechanism supports greatly the animal adaptation to the environment, once the animal itself is able to formulate behaviors each time more suitable to pleasant or risky situations.

In [21], it is proposed that learning consists of establishing a connection between a response and production of a pleasant situation. He also establishes the foundations of operant conditioning through the study of learning via rewarding consequences and formulates some learning "laws". The effect law establishes that a response is strengthened if pleasure is followed; otherwise it is weakened in the case pain is followed. The repetition law establishes that a stimulus-response pair will be retained for longer time as this pair is more repeated and connected with a reward.

Skinner [29] argues the learning takes place according to these two types of mechanisms, namely, respondent and behavior .While the respondent behavior (classical conditioning) is controlled by a precedent stimulus (e.g., a bell sound and presentation of food simultaneously), operant behavior (operant conditioning) is controlled by its consequences – stimuli (reinforcement) that succeed the responses (e.g., when a dog reaches a food the stimulus given by this food reinforces the seeking action for more food). The importance of reinforcement and of repetitive experiences is also highlighted by Skinner.

For Skinner, learning is defined as a change in the probability of presenting a response, usually evoked by operant conditioning - association of a response (action) and a reinforcement. When the reinforcement is positive (a reward), the probability of presenting the respective response in the future is increased. On the other hand, if the reinforcement is negative, the respective response will be less often in the future (its probability of coming up is decreased), constituting a punishment.

More details about operant conditioning are given by Baum in [12]: he presents four types of relations: positive reinforcement, negative reinforcement, positive punishment and negative punishment.

2.4 Neuronal Group Selection

The theory related to neuronal group selection developed by G. Edelman tries to understand the phenomenon of intelligence from the investigation of biological nervous systems [10].

The principles involved are based in aspects of the Darwin's natural selection theory. Edelman suggests that competitive interactions among neurons contribute to the selective formation of connections and groupings, generating a progressive specialization in the function of involved neural structures. Three stages in the organization of the nervous system are considered in his theory: phase of selection by development, phase of selection by experience and phase of formation of reentrant mappings.

The phase of selection by development is determined by phylogenetic factors and it is responsible for the manifestation of innate behaviors in the individual. The following phase (selection by experience) is predominated by learning processes through individual's interactions with the environment. In this sense, neuronal groups partially developed on the previous phase are now specialized, having their synaptic connections strengthened or weakened. The last phase (formation of reentrant mappings) consists of interconnections between several neural repertoires with distinct functionalities. More complex functions (than in the previous phases) are developed in this phase. Additionally, the individual is more adaptable to environmental changes in this last stage.

3 Autonomous Navigation System

3.1 Introduction

Biologic neuronal systems hold innate (instinctive) mechanisms (e.g., sucking, grasping; and hunger and fear sensations) to support the emergence of specific behaviors acquired from the interactions with the environment [8 and 9]. Innate behaviors are essential to the integrity and development of the animal, although many can be evoked only in a short period of time after birth. They are not associated to particular life experiences established with the environment, but are a priori incorporated in the nervous system, during the epigenesis.

Considering biologic systems as a reference, innate (pre-incorporated) behaviors hold a function equally preponderant in intelligent autonomous systems [10]. Under this interpretation, the design of the proposed system is conducted, either referring to its learning scheme or to its architecture. This chapter describes the autonomous navigation system presented in [28]. The robot model is presented in Section 3.2. Sections 3.3, 3.4 and 3.5 describe system's architecture, reasoning and learning, respectively. Current navigation systems limitations are presented in Section 3.6.

3.2 Robot Model

Repulsive and attractive objects compose the navigation environment. Each object is of a particular color, coherently with its respective class (for instance, repulsive and attractive objects may exhibit blue and green colors, respectively). Obstacles also belong to the repulsive class.

The robot model is shown in Fig. 3.1. The robot interacts with the environment by distance, color and contact sensors; and by one actuator system that controls the movement inclusive the direction. The sensors are organized in 67 predefined positions in front of the robot (distributed at the interval of –90º a +90º). Thus, there are three distinct sensors for each one of the 67 predetermined positions. Each sensor provides specific information related to the closest object to the robot on the direction of the sensor (see Fig. 3.1). A distance sensor detects the distance between the robot and the closest object and emits values in the interval [0, 1]. Color sensors also emit values in the interval [0,1] – the conversion is made by normalization of the "hue" component of the HSV system (Hue, Saturation, Value). Binary contact sensors detect the touches of the robot with any object in the environment.

The task of the robot is to reach attractive objects establishing a trajectory free of collisions with obstacles. When the robot reaches an attractive object (a possible target), this object is removed from the environment (simulating a collector robot).

The velocity of the robot is constant during the navigation (0.28 distance units per iteration). At each iteration, the robot is able to execute no direction adjustment greater than 15º.

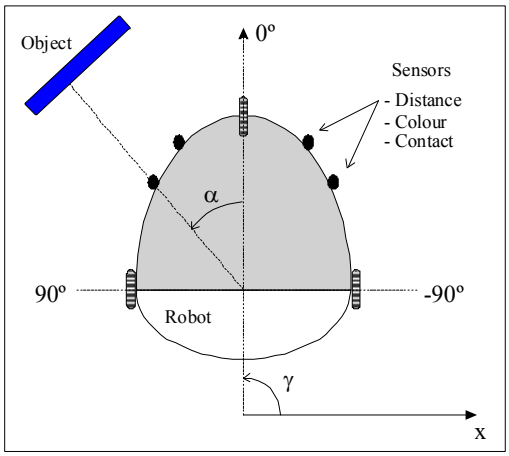


Figure 3.1: Robot model.

3.3 Architecture

The intelligent autonomous system corresponds to a neural network arranged in three layers (Fig. 3.2). At the first layer, there are two neural repertoires: Proximity Identifier repertoire (PI) and Color Identifier repertoire (CI). Both repertoires receive stimuli from contact sensors. PI and CI repertoires receive stimuli from the two other sensor fields, distance sensors and color sensors, respectively. The second layer comprehends two neural repertoires: Attraction repertoire (AR) and Repulsion repertoire (RR). Each one establishes connections with both networks at the first layer, as well as with contact sensors. The actuator network, connected to AR and RR repertoires, determines the direction adjustments.

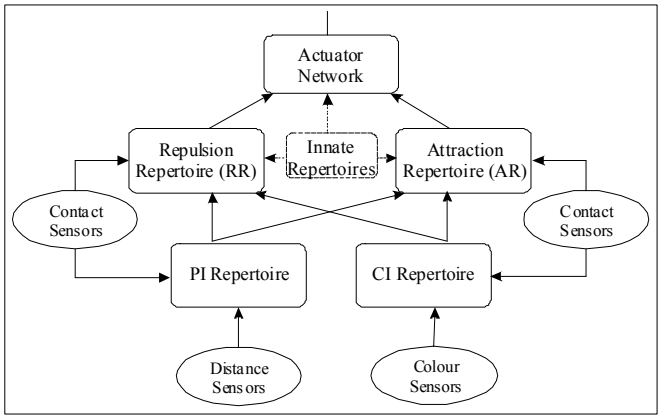


Figure 3.2: Autonomous system architecture.

PI and CI repertoires: The architecture of the neural network that composes each repertoire is presented in Fig. 3. Each column of neurons is fixed, topological, and one-dimensional structured. There is no spatial influence among neurons of different columns. Each neuron of a column establishes connections with a group of sensors defined according to a specific probability distribution (Fig. 3.3).

AR and RR Repertoires: A unique layer of neurons forms AR and RR neural networks. The size of each network corresponds to the number of columns in IP or IC networks. Each neuron in this layer connects to every neuron in the corresponding column of IP and IC networks, respectively (Fig. 3.4).

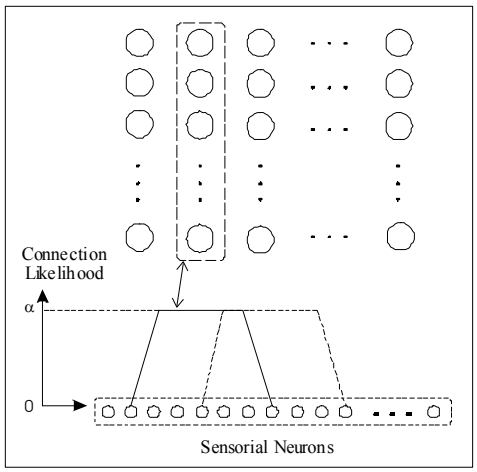


Figure 3.3: PI and CI architectures.

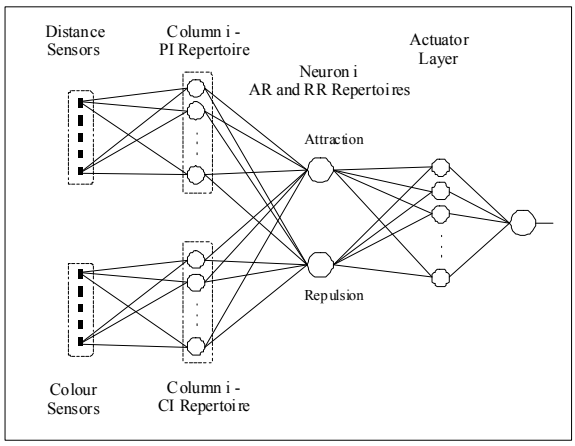


Figure 3.4: Architecture overview showing only one PI and CI column as well as the corresponding AR and RR neuron.

Actuator Network: There are two layers in the actuators network. Each neuron in the first layer is associated to a fixed and predefined direction adjustment value. Each neuron receives stimuli from neurons in RA and RR networks and it is also connected to the unique neuron in the output layer (Fig. 3.5).

Innate Repertoire: They are connected to the contact sensorial field (not shown in Fig. 3.1 to raise clarity), and to the AR, RR, and actuator networks. The role of innate repertoires is related to: generation of (attractive or repulsive) reflexes when the robot touches an object; and activation of specific parts of system's neural network when a learning process is fired.

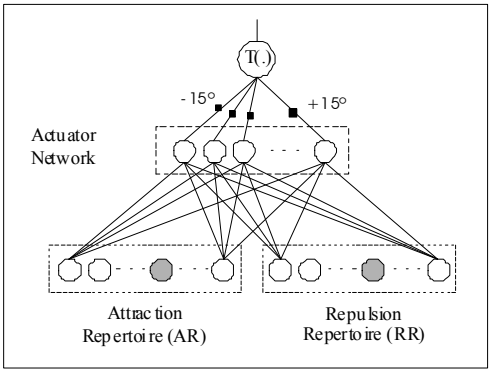


Figure 3.5: Actuator Network.

3.4 Reasoning

Different neuron models are adopted to design the intelligent autonomous system. The particular characteristics determine which neural network they compose.

Repertoires PI and CI: The output of the j^{th} neuron at the k^{th} column of PI repertoire, at iteration n, is defined as in (1):

$$y_j^k(n) = \begin{cases} u_j^k(n) & \text{if } j = i^k(\mathbf{x}^k(n)) \\ 0 & \text{if } j \neq i^k(\mathbf{x}^k(n)), \end{cases} \tag{1}$$

where:

$i^k(\mathbf{x}^k(n))$ is the winner neuron;

$\mathbf{x}^k(n) = [x_1, x_2, ..., x_m]^T$ is the input vector (distance to objects);

$u_j^k(n)$ is the activation potential (defined in Section *3.5. Learning*);

$k = 1,2,...q$ (q: numbers of columns).

The winner neuron $i^k(\mathbf{x}^k(n))$ is defined by:

$$i^k(\mathbf{x}^k(n)) = \arg\min_j \left\| \mathbf{x}^k(n) - \mathbf{w}_j^k(n) \right\|, \; j = 1,2,...,l, \tag{2}$$

where:

$\mathbf{w}_j^k(n)$ is the vector of synaptic weights;

l is the number of neurons in each column.

Besides $u_j^k(n)$, each neuron exhibits another inner parameter: $e_j^k(n)$ (degree of activity), defined as:

$$e_j^k(n+1) = \begin{cases} e_j^k(n) + \varphi e_j^k(n) & \text{if } j = i^k(\mathbf{x}^k(n)) \\ e_j^k(n) - \sigma e_j^k(n) & \text{otherwise,} \end{cases} \tag{3}$$

where:

φ is the gain factor;

σ is the loss factor.

Equation (4) defines the output of the j^{th} neuron at the k^{th} column of the CI repertoire:

$$y_j^k(n) = \begin{cases} 1 & \text{if } j = i^k(\mathbf{x}^k(n)) \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $i^k(\mathbf{x}^k(n))$ is defined in the same manner as in (2).

AR and RR Repertoires: The output of the j^{th} neuron of both AR and RR repertoires is given by:

$$y_j(n) = \begin{cases} 0 & \text{if } s_j <= \beta \\ \alpha(s_j - \beta) & \text{if } \beta < s_j < (\beta + 1/\alpha) \\ 1 & \text{if } s_j >= (\beta + 1/\alpha), \end{cases} \tag{5}$$

where:

$s_j = \mathbf{x}_j(n) \cdot \mathbf{w}_j(n)$ is the inner product between the input vector and the synaptic weights vector;

α and β are constants;

$j = 1, 2, ..., q$ (q: size of the neural network).

Actuator Network: The output of the j^{th} neuron at the first layer of the actuator network is given by:

$$y_j(n) = g\left(\sum_{i=1}^{2q} x_{ji}(n) w_{ji} \right), \tag{6}$$

where $g(.)$ is the hyperbolic function.

Observe that there are $2q$ inputs corresponding to the total of $2q$ neurons in AR and RR networks.

The output of the actuator network has the following expression:

$$y = \sum_{j=1}^{r} w_j \bar{x}_j \,,\ \bar{x}_j = x_j / \sum_{p=1}^{r} x_p \text{, and } w_j = 15(2j - r)/r \,, \tag{7}$$

where r is the number of neurons in the first layer.

The synaptic weight w_j is constant and equal to some value in the interval [-15, 15]; and this interval corresponds to the possible adjustments on the movement direction.

3.5 Learning

The learning procedure develops according to the classic reinforcement learning theory and is based on the activity of innate repertoires. The main concepts involved are inspired by the biological counterpart [9]. Once a contact with objects is detected, the innate repertoire generates one of the innate behaviors (attraction or repulsion) and it starts the learning process. The learning mechanisms, which deal with adjustments on the synaptic weights, are specific for each neural repertoire and are described next.

PI and CI Repertoires: Each time a contact occurs (between the robot and objects) only the sensor (contact sensorial field) closest to the point of the contact detects it. The corresponding stimulus defines a unique column in CI and in PI networks (consider the k^{th} column). The adjustment occurs on the synaptic weights from the selected column (there is a spatial correspondence between positions of contact sensors and columns in CI, as well as in PI networks).

The following procedures model the synaptic adjustment mechanisms on the PI repertoire:

1- Competition by Similarity: calculate the winner neuron $i(n)$ given by:

$$i(n) = \arg\min_j \left\| \mathbf{x}(n) - \mathbf{w}_j(n) \right\|,\ j = 1,2,...,l. \tag{8}$$

2- Adjustment:

i) Adjust the synaptic weight vectors of all neurons by applying (9):

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta h(i(n), j)(\mathbf{x}(n) - \mathbf{w}_j(n)), \tag{9}$$

where:

η is the learning rate;

$h(i(n), j) = \exp\left(-(i(n) - j)^2 / l\right)$ is the neighborhood function.

ii) Adjust the following parameters for $j = 1,2,...,l$:

$$u_j(n+1) = u_j(n) + \eta h(i(n), j)(1 - u_j(n))e_j(n)/3;$$
$$e_j(n+1) = 1. \tag{10}$$

Note that $u_j(n)$ is the activation potential in the neuron reasoning model and, $e_j(n)$ is also adjusted during reasoning, at each iteration n (see Section *C. Reasoning*).

The learning algorithm for the CI repertoire is described next:

1- Competition by Similarity: calculate the winner neuron $i(n)$ given by:

$$i(n) = \arg\min_j \left\| \mathbf{x}(n) - \mathbf{w}_j(n) \right\|,$$
$$j \in \left\{ k / k \in \{1,2,...,l\} \text{ and } \left\| \mathbf{x}(n) - \mathbf{w}_k(n) \right\| < m_k(n) \right\}. \tag{11}$$

where $m_k(n)$ is the acceptance parameter of the neuron k.

2- Adjustment: Adjust the synaptic weight vector and the acceptance parameter:

$$\mathbf{w}_{i(n)}(n+1) = \mathbf{w}_{i(n)}(n) + \eta(\mathbf{x}(n) - \mathbf{w}_{i(n)}(n))$$
$$m_{i(n)}(n) = \left\| \mathbf{x}(n) - \mathbf{w}_{i(n)} \right\| \tag{12}$$

where η is the learning rate.

AR and RR Repertoires: The synaptic weights of the j^{th} neuron (AR or RR) correlated to the inputs connected to the PI repertoire are all unitary (constant values). Differently, the synaptic weights associated with the inputs connected to the CI repertoire are adjustable.

Consider that neurons at the k^{th} column of the CI repertoire have adjusted their synaptic weights. Then, the learning mechanism acts on the k^{th} neuron of the AR network if the innate behavior generated is of attraction type. Otherwise, it acts on the k^{th} neuron of the RR repertoire.

The synaptic weights adjustment is given by (13):

$$\mathbf{w}_k(n+1) = \mathbf{w}_k(n) + \eta \mathbf{x}(n)(1 - \mathbf{w}_k(n)), \tag{13}$$

where η is the learning rate.

Actuator Network: Only the synaptic weights of the neurons at the first layer of the actuator network are adjusted. The innate repertoire reproduces some of the innate behaviors (attraction or repulsion) when a contact between robot and some object occurs. To accomplish that, it coherently activates neurons (in the first layer of the actuator network) to generate the respective behavior. The neuron activation is generated by the following equation:

$$y_j(n) = h(j, v)$$

where:

$y_j(n)$ is the output of the j^{th} neuron; j=1,2,…,r;

v is the neuron with strongest activation in the actuator network;

$h(j,v)$ is the function that generates the output in the actuator network.

The neuron v is calculated by the following algorithm:

if the contact is attractive: $v = s_i.r/m$;

if the contact is repulsive: $v = \begin{cases} s_i.r/m + r/2 & \text{if } s_i/m < 1/2 \\ s_i.r/m - r/2 & \text{otherwise} \end{cases}$.

where:

s_i is the index of the activated sensor (in the interval [0, m]);

m is the number of contact sensors;

r is the number of neurons in the first layer of the actuator network;

The function $h(j,v)$ is given by ($\sigma = r/5$):

$$h(j, v) = \exp(-(j - v)^2 / (2\sigma^2))$$

The adjustment of the synaptic weights of the j^{th} neuron at the first layer of the actuator network is expressed in (14):

$$w_{ji}(n+1) = w_{ji}(n) + \eta y_j x_i(n)(1 - w_{ji}(n)), \tag{14}$$

where:

$y_j(n)$ is the output of the j^{th} neuron;

$x_i(n)$ is the i^{th} input;

η is the learning rate.

3.6 Configuration of Simulation Experiments

Every experiment in this work is executed on a computational environment provided with suitable simulation tools developed specifically to test and analyze autonomous controllers (see Appendix B).

Figures (generated from simulations) show a global view of the navigation environment at the last iteration (end of simulation) and, in particular, the robot trajectories (a black line). The robot is represented by a (green) semicircle. Repulsive and attractive objects are represented by dark (blue) and clear (yellow) polygons, respectively (due to edition limitations it can be impossible to exhibit color objects). Repulsive contacts (or collisions) are marked with clear (green) circles. Attractive contacts are marked with black (red) circles.

For all the experiments whose environment contains attractive objects, there is replacement of these attractive objects after the robot captures all them. For instance, if there are 5 attractive objects in the environment, then after the robot captures the last (fifth) one, the other four objects appear at their respective positions again (but not the last captured). This fifth object will reappear at the next time after the other four objects are captured. Thus, there will be always no more than 4 objects in the environment (except at the beginning of the simulation). In this way, it is possible to measure the performance of the autonomous system with respect to attractive and repulsive contacts during long simulations.

For all simulation experiments in this work, distance sensors are considered to be noisy (Gaussian noise is generated and inserted in the sensors value - $\sigma \cong$ size of the robot). Simulations in [28] did not consider noise on sensors. Thus, it can be verified that the autonomous controller is also robust to noisy sensor information.

Concerning simulations for this chapter, PI and CI networks configurations are structured in 25 columns with 12 neurons each;

$e_j(n) \in [0, 2]$;

$\varphi = 0.028$; $\sigma = 0.013$;

$e_j(0) = 1$; $u_j(0) = 0.06$; $m_j(0) = 50$.

Regarding RR and AR networks: α =1.3 and β = 1.1. Every synaptic adjustment uses learning rate $\eta = 0.4$. Weights in PI and CI are initialized with random values. In the case of AR, RR and actuator network (first layer), the random values are restricted within the interval [0, 0.07]. The slope parameter for the hyperbolic function in (6) is 2.5. This configuration of parameters are identical to the one presented in [28].

3.7 Simulations and Current Limitations

In [28], simulations results show that the autonomous navigation system is able to learn a suitable navigation strategy for avoiding repulsive objects (possible obstacles) and for capturing attractive objects (possible targets). However, the training environment chosen for those simulations has special characteristics: there are obstacles inside the environment. This characteristic makes the learning easier once the autonomous system can interact diversely with the environment (i.e., more distinct contacts are made possible).

Besides, extensive simulations are needed so that system's performance is accurately measured. For instance, there are no performance experiments with respect to the number of collisions with

obstacles and the number of target captures. The number of iterations for each simulation is also not taken into account in [28].

In Fig. 3.6, it is possible to observe a case in which the mobile robot is trapped in an endless unsuitable behavior. The environment is not considered rich (i.e., with sufficient number of obstacles for appropriate learning) such that system's learning progresses adequately. It contains two attractive objects, each one situated at the end of the large corridor. For 38 simulations (each simulation lasts for 30.000 iterations) and three different initial robot positions (see Fig. 3.7), there was only one case in which the robot was able to capture both attractive objects in the environment. Although, special initial robot positions in this environment can influence positively on the autonomous system's learning (the initial positions such that several collisions on the left part of the robot are followed by several collisions on its right part).

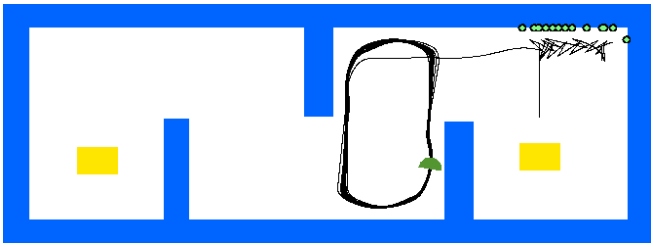
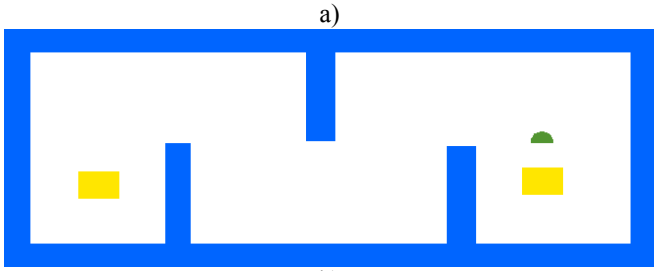
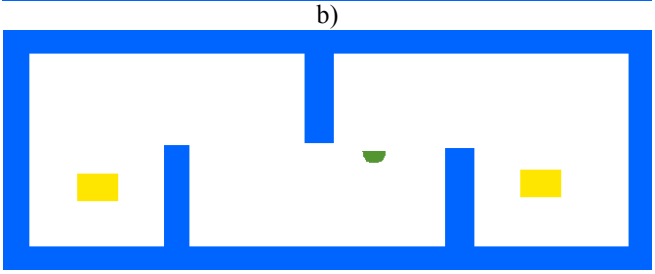


Figure 3.6: Autonomous navigation system generates an endless cyclic trajectory in a simple environment without obstacles inside.

a)



b)



c)

Figure 3.7: Three different initial robot positions for environment "large corridor".

In Fig. 3.8, it is shown a rich environment in which there are several attractive and repulsive objects. Such environment is ideal for diverse learning so that endless cyclic trajectories are avoided in the beginning of experiments. For this particular environment, each attractive object disappears (it is captured by the robot) after the robot touches it twice. The intention for this is to accelerate the attractive behavior generation.

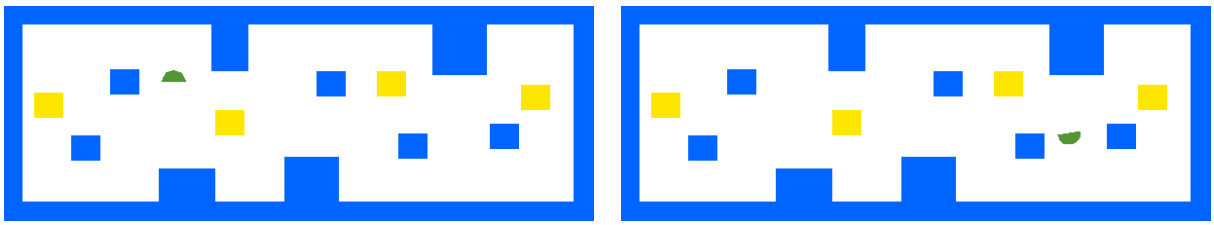


Figure 3.8: Two different initial robot positions for a diverse environment with many obstacles and targets.

Eight long simulations (with 150.000 iterations each) were run using the environment in Fig 3.8 (and two different initial robot positions). The worst case is shown in Fig. 3.9 (a): the simulation finishes with the robot trapped in one of the corners of the environment. The respective learning evolution graphic (that shows the (accumulated) number of attractive and repulsive contacts over time) is shown in Fig. 3.9 (c). The blue curve corresponds to the number of repulsive contacts (or collisions) and the yellow curve corresponds to the number of attractive contacts (possibly target captures).

It is possible to note that the blue curve increases a lot after one hundred thousand iterations, representing the moment in which the robot gets trapped in the corner. The best case is shown in Fig. 3.9 (b). Its respective learning evolution graphic (Fig. 3.9 (d)) shows that the number of captures with time is more satisfying than in the worst case (yellow curve increases with time) and the number of collisions nearly stabilizes (blue curve). Although, the best case has occurred only once from eight simulations. Other simulations are similar to the worst case or finish with constant blue and yellow curves (cyclic unsuitable trajectories).

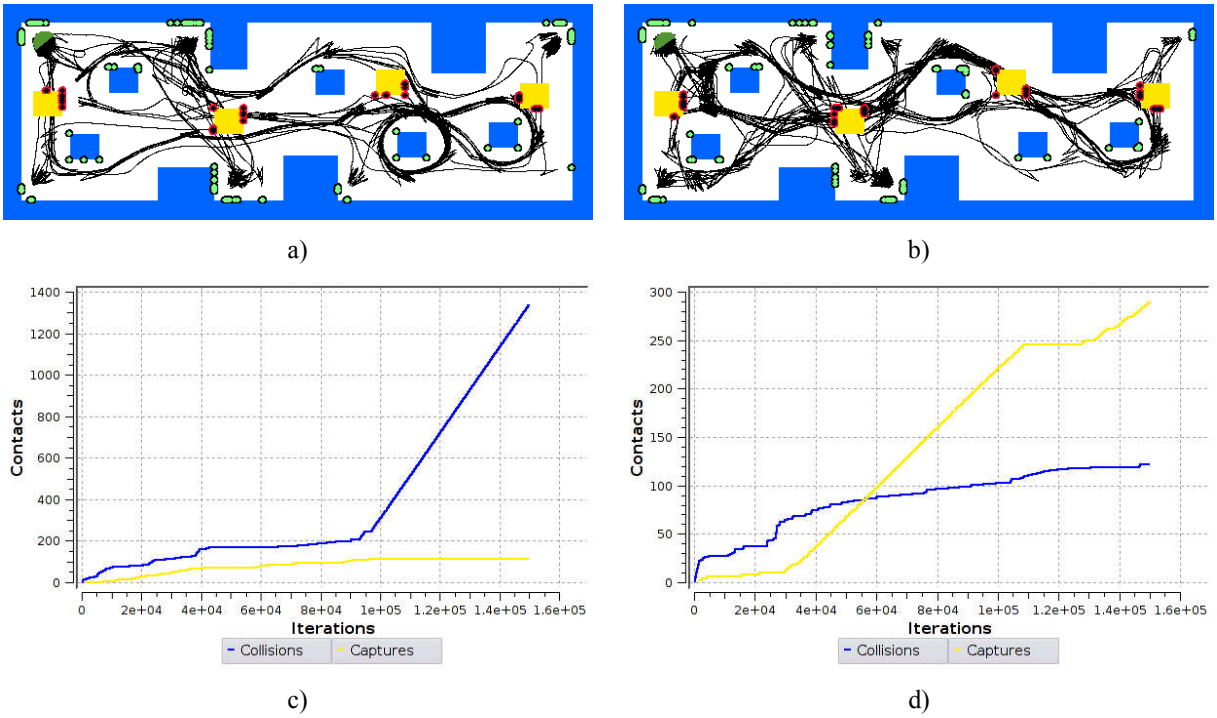


Figure 3.9: Simulation environments and respective learning evolution graphics; (a) Worst case: the robot finishes trapped in the upper left corner of the environment; (b) Best case: the robot learns to navigate in the environment seeking attractive objects most of the time, but the corner configuration is still a problem; (c) Learning evolution graphic showing the accumulated number of attractive (yellow curve) and repulsive (blue curve) contacts for simulation (a); (d) Learning evolution graphic for simulation (b).

It seems clear that the current autonomous navigation system (from [28]) has important limitations. Firstly, the type of training environments and initial robot positions are determining for the system's learning and adequate (attractive and repulsive) behavior generation. For instance, the autonomous navigation system generates an endless cyclic trajectory when the training takes place in a simple environment (Fig. 3.6) and the initial robot position is not carefully chosen.

Secondly, the number of collisions does not stabilize for most of the simulations in certain environments (although the robot can navigate in the environment and seek targets). One such example is shown in Fig 3.9 (a) and (c): the (blue) curve for collisions does not stabilize.

4 Learning by Punishment

4.1 Introduction

The autonomous navigation system previously presented only learns by classical conditioning. As it is shown in Section 3.7 from Chapter 3, the system has some limitations with respect to the initial configuration of experiments: the type of environment and initial position of the robot. Depending on the choice of these two variables (e.g., there are no repulsive objects inside the environment and the initial robot position is unfavorable for learning an efficient navigation strategy), the system can be trapped in an endless cyclic trajectory, making the robot unable to exploit the environment.

This chapter describe a new mechanism based on negative reinforcement from operant conditioning [29], aiming at the solution for the aforementioned problem: endless cyclic trajectories. The negative reinforcement rule decreases the probability that the system shows the unfavorable behavior in the future by punishing the neurons responsible for the bad behavior. This type of negative reinforcement has been used in autonomous navigation systems in order to improve performance with respect to number of collisions [4].

4.2 Monotony Detection

The first step towards solving the problem of endless cyclic trajectories is the addition of a module in the system that detects endless cyclic trajectories. The most suitable and natural way for detecting this situation is with a declarative memory system. This memory system (possibly composed of Hopfield neural networks [31]) should store visual patterns (e.g., distance and color patterns) and then compare the current input pattern to the stored ones. Thus, the detection of the unfavorable situation is achieved in a suitable way and following the particularities of biological systems.

However, the aforementioned mechanism is not used in this work. Instead, the proposed model for detecting endless cyclic trajectories is composed of a simple set of rules. It consists of an innate module in the system that detects if the robot has not experienced any repulsive or attractive contact for a certain time interval. Furthermore, this time interval (in iterations and defined below) is increased each time a monotony event is detected:

$$\lambda(n+1) = \lambda(n) + 0.2\lambda(n).$$

Thus, the autonomous system has an internal state for monotony. A similar mechanism for detecting monotonies is also used in an autonomous navigation system described in [32].

This solution (called monotony detection) actually detects endless cyclic trajectories, but there are drawbacks with this approach. For instance, if the robot has not had any contacts with any (attractive or repulsive) objects for a long time interval, but it is not trapped in an unfavorable cyclic trajectory, the monotony detection module would still detect the monotony situation. The use of the memory system as a detection method for endless cyclic trajectories would require further research efforts and it is not the aim of this work.

The monotony detection mechanism is considered as a part of the innate repertoires showed in Fig. 3.2. It is connected to PI and RR repertoires (i.e., the monotony detection can fire learning events in PI and RR repertoires).

4.3 Proximity Identifier (PI) Repertoire

The architecture of PI repertoire is identical to the one presented in Fig. 3.3 and Fig. 3.4. On the other hand, PI repertoire reasoning and learning are changed.

PI Repertoire Reasoning: The output of the j^{th} neuron at the k^{th} column of PI repertoire, at iteration n, is defined as in (15):

$$y_j^k(n) = \begin{cases} u_j^k(n) f(\phi, n, j) & \text{if } j = i^k(\mathbf{x}^k(n), \Theta) \\ 0 & \text{if } j \neq i^k(\mathbf{x}^k(n), \Theta), \end{cases} \tag{15}$$

where:

$i^k(\mathbf{x}^k(n), \Theta)$ is the winner neuron;

$\mathbf{x}^k(n) = [x_1, x_2, ..., x_m]^T$ is the input vector (distance to objects);

$u_j^k(n)$ is the activation potential (defined in (20));

$f(\phi, n, j)$ is a modulator (Gaussian) function;

ϕ is the dispersion parameter for the modulator function;

$k = 1,2,...q$ (q: numbers of columns).

The modulator function $f(\phi, n, j)$ is defined next:

$$f(\phi, n, j) = \exp\left((0 - \left\|\mathbf{x}^k(n) - \mathbf{w}_j^k(n)\right\|)^2 / \phi\right) \tag{16}$$

This function outputs high values (close to 1) if the input pattern is very close to the synaptic weights vector (stored pattern) and low values if they are not so similar. Naturally, the behavior of this function is also influenced by the dispersion parameter ϕ, which alters the slope of the function.

The winner neuron $i^k(\mathbf{x}^k(n), \Theta)$ is defined by:

$$\begin{aligned} & i^k(\mathbf{x}^k(n), \Theta) = \arg\min_j \left\|\mathbf{x}^k(n) - \mathbf{w}_j^k(n)\right\|, \\ & j \in \left\{ P \cup \left\{ u / u \in \overline{W}^k \text{ and } P = \{\} \right\} \right\}, \text{where } P = \left\{ u / u \in W^k \text{ and } \left\|\mathbf{x}^k(n) - \mathbf{w}_u^k(n)\right\| < \Theta \right\} \end{aligned} \tag{17}$$

where:

$\mathbf{w}_j^k(n)$ is the vector of synaptic weights;

W^k is the set of neurons in column *k* that have already been winners;

Θ is the acceptance parameter for neurons in W^k.

The neurons in PI repertoire pertaining to W^k are called proud neurons once they are fired only if the input pattern is sufficient similar to the synaptic weight vector. Otherwise, another neuron pertaining to $\overline{W}^k$ is fired.

The current neuron model does not have the parameter of degree of activity $e_j^k(n)$ anymore, presented in Chapter 3.

PI Repertoire Learning: The learning firing for PI repertoire and the selected column (consider the k^{th} column) are defined in the same way as explained in Chapter 3, except for the new restriction that only repulsive contacts detected by innate repertoires generate learning events in PI repertoire (that is, an attractive contact generates learning events in all system repertoires except in the PI repertoire). This restriction is necessary for learning suitable repulsive behaviors (to obstacles) and consequently a better exploring behavior.

The following procedures model the synaptic adjustment mechanisms on the PI repertoire:

1- Competition by Similarity: calculate the winner neuron $i(n)$ given by:

$$\begin{aligned} &i^k(n,\mathrm{O}) = \arg\min_j \left\| \mathbf{x}^k(n) - \mathbf{w}_j^k(n) \right\|, \\ &j \in \left\{ P \cup \left\{ u / u \in \overline{W}^k \text{ and } P = \{\} \right\} \right\}, \text{where } P = \left\{ u / u \in W^k \text{ and } \left\| \mathbf{x}^k(n) - \mathbf{w}_u^k(n) \right\| < \mathrm{O} \right\}. \end{aligned} \tag{18}$$

where:

O is the acceptance parameter for the proud neurons in W^k during learning;

other variables are defined analogously to (17);

2- Adjustment:

i) Adjust the synaptic weight vector of the winner neuron by applying (19):

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta_j(n)(\mathbf{x}(n) - \mathbf{w}_j(n)), \tag{19}$$

where: $\eta_j(n)$ is the learning rate of the winner neuron defined by $\eta_j(n) = 0.6\eta_j(n-1)$;

ii) Adjust the following parameter for the winner neuron *j*:

$$u_j(n+1) = u_j(n) + \hat{\eta}(0.6 - u_j(n)) \tag{20}$$

where: $\hat{\eta}$ is a fixed learning rate value.

Note that $u_j(n)$ is the activation potential in the neuron reasoning model.

With the new modifications, PI repertoire can learn different proximity patterns for a contact detected by the same contact sensor. The learning is more accurate because of the addition of the concept of proud neurons: the new pattern is inserted only if it is sufficiently distinct from the stored pattern; otherwise, an synaptic adjustment on the proud neuron is accomplished. The reasoning is also changed: proud neurons are fired only if the input pattern is sufficiently similar to the stored pattern. Additionally, the similarity between the input pattern and the stored pattern (i.e., synaptic weights vector) also influences the output of PI repertoire neurons (this is caused by the modulator function).

Mechanisms of stabilization of number of collisions (for risky configurations), described in the next chapter, exploit the modifications accomplished in the PI repertoire.

4.4 Repulsion Repertoire

The architecture of RR is identical to the one presented in Chapter 3 and detailed in Fig. 3.3 and Fig 3.4. RR reasoning and learning are also not changed (defined by (5) and (13) respectively), except for a new inner parameter (degree of activity) that is adjusted at reasoning time (i.e., at each iteration). The degree of activity is defined as:

$$e_j^k(n+1) = \begin{cases} e_j^k(n) + \varphi e_j^k(n) & \text{if } y_j > \kappa; \\ e_j^k(n) - \sigma e_j^k(n) & \text{otherwise,} \end{cases} \tag{21}$$

where:

φ is the gain factor;

σ is the loss factor;

κ is a very low value such that any neuron that is slightly activated (i.e., has output different from zero) has its degree of activity increased.

The degree of activity has an important role in learning by punishment mechanism, described in Section 4.6.

4.5 Innate Reflexes

When the robot touches an object, there is a corresponding reflex that is generated (by the innate repertoire of the autonomous system) according to the point of contact and the nature of the contact (attractive or repulsive).

For instance, if there is a frontal collision with an obstacle, the reflex will be stronger (outward the obstacle) than if the collision is laterally located. In the case of attractive contacts, the opposite takes place: lateral contacts generates stronger reflexes (towards the target) than frontal contacts.

The innate repertoire of the original autonomous system works in the way explained above. Though, it does not generate reflexes that results in direction adjustments higher than 15^{o}.

From now on, there is an additional factor of randomness in the generation of reflexes. The direction adjustment can be 50^{o} higher for reflexes (this additional factor is randomly generated). In this manner, the robot can exploit navigation experiences more freely during the simulation once less consecutive contacts are made possible. Additionally, the autonomous system is able to escape environment corners after few collisions (given that the robot is trapped in the corner configuration).

4.6 Punishment of Neurons

After presenting the improvements made in PI and RR repertoires (on above sections), the learning by punishment is described in this section. The learning rule is based on negative reinforcement (from operant conditioning [29]), that is, when an aversive (negative) stimulus (in this case the monotony stimulus) that follows an behavior (response) is presented to the autonomous system, the former (behavior or response) is debilitated by punishment. In this case, the behavior corresponds possibly to endless cyclic trajectories (detected by system's innate mechanisms – see Section 4.2 *Monotony Detection*).

Endless cyclic trajectories are caused by an unsuitable activation pattern in RR network. For instance, if the robot collides several times on its left part, but there is no collision on its right part, then probably it will avoid only repulsive objects detected on its left part (generating a cyclic behavior). This situation is possible to take place because there is a spatial relationship in the system's architecture: the connections between sensors and PI and CI repertoires are spatially distributed – see Fig. 3.3.

One could propose a symmetry mechanism in the learning procedure so that each collision on one side generates learning events on both sides of the neural network architecture. However, there are no evidences that this mechanism is present in biological systems (for instance, consider the reactive skill of handling an object (with the hand): it is possible to realize that each hand has got particularities for the task that are determined by each hand's experience).

Once it is known that the activation in RR neurons are an indication for the unsuitable behavior, the next step is to find out which neurons are contributing to the unfavorable trajectory. And this is the reason for adding the inner parameter degree of activity into RR neurons (defined in (21)). This degree of activity makes possible to know which neurons were being activated in the past, and consequently, the neurons that were contributing to the cyclic behavior.

The learning rule activated after the presentation of the aversive stimulus (monotony stimulus) is composed by a set of adjustments presented next:

1 – Activation potential adjustment in PI repertoire neurons:

$$u_j^k(n+1) = u_j^k(n) + \eta(e_k(n)/5)(0 - u_j^k(n)),$$
$$k \in \{r / e_r > \tau \text{ and } r = 1,2,...,q\}, \qquad j = 1,2,...,l \tag{22}$$

where:

u_j^k is the activation potential of the j^{th} neuron at the k^{th} column in PI repertoire;

$e_k(n)$ is the degree of activity of the k^{th} neuron in RR repertoire;

q is the size of RR neural network;

τ is a threshold for selecting the neurons that are adjusted during learning and

η is the learning rate.

2 – Weight adjustment in RR neurons:

$$w_k = w_k + \eta(e_k(n)/5)(0 - w_k),$$
$$k \in \{r / e_r > \tau \text{ and } r = 1,2,...,q\}, \tag{23}$$

where:

w_k is the adjustable synaptic weight of the kth neuron in RR neural network (connected to CI repertoire);

other parameters are defined analogously to (22).

3- Parameter Adjustment in RR repertoire:

$$\beta(n) = \beta_1 .$$

The above adjustment influences the activation function of neurons in RR repertoire (defined in (5)). The value β_1 is such that the activation of neurons is debilitated (what increases the probability for future interactions with the environment). In the next iterations that follow the monotony event, $\beta(n)$ reaches gradually the ordinary value β_0:

$$\beta(n) = \beta(n) + \eta_\beta \beta_0,$$

where η_β is the update rate.

4.7 Simulation Results

Simulations results are presented in this section in order to confirm the initial objectives: make possible that the robot is able to exploit the environment and achieve its goals (i.e., seek targets and avoid repulsive contacts), without being indefinitely trapped in unsuitable trajectories.

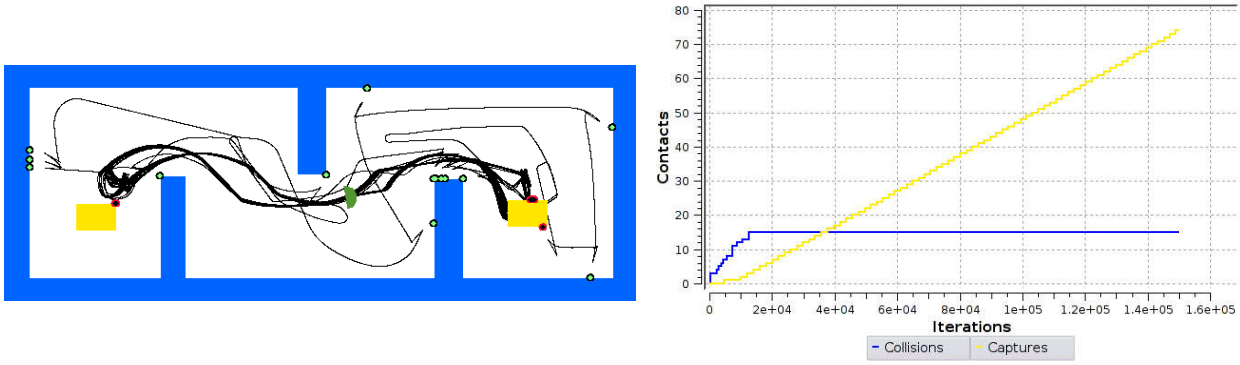
For this purpose, the environment shown in Fig. 3.7 and three different initial robot positions (also shown in the same figure) are considered for the following simulation experiments. This choice is motivated by simulation results shown in Chapter 3 – Section 3.7 and Fig. 3.6: the robot gets easily confined in an unsuitable trajectory for such environment.

Explanations presented in Section 3.6 *Configuration of Simulation Experiments* are also valid for this chapter. Concerning parameters configuration,

η=0.45; τ=0.1; β_0=1.01; β_1=2.5; η_β=0.002;	(Section 4.6 *Punishment of Neurons*)
$\kappa = 0.001$; $\varphi = 0.02$; $\sigma = 0.0005$; $e_j(0)$=1; $e_j(n) \in [0.1, 5]$;	(Section 4.4 *Repulsion Repertoire*)
$\Theta = 0.63$; $O = 0.26$; $\hat{\eta}$=0.7; $\phi = 0.5$; $\eta_j(0) = 0.7$; $u_j(0) = 0.06$;	(Section 4.3 *Proximity Identifier Repertoire*)
$\lambda(0) = 2300$; $\lambda(n) \in [2300,8000]$;	(Section 4.2 *Monotony Detection*)
$m_j(0) = 50$;	(*Color Identifier Repertoire*)

PI and CI networks configurations are structured in 22 columns (q) with 12 neurons (l) each; Regarding RR and AR networks: α =1.3 and $\phi(0)$ = 1.1. Synaptic adjustments in AR and RR neural networks use, respectively, learning rates $\eta' = 0.8$ and $\eta'' = 0.65$. Synaptic adjustments in CI network (defined by (12)) are now defined like in (19) with the varying learning rate $\eta_j(0) = 0.9$. Weights in PI and CI are initialized with random values. In the case of AR, RR and actuator network (first layer), the random values are restricted within the interval [0, 0.07]. The slope parameter for the hyperbolic function in (6) is 2.5.

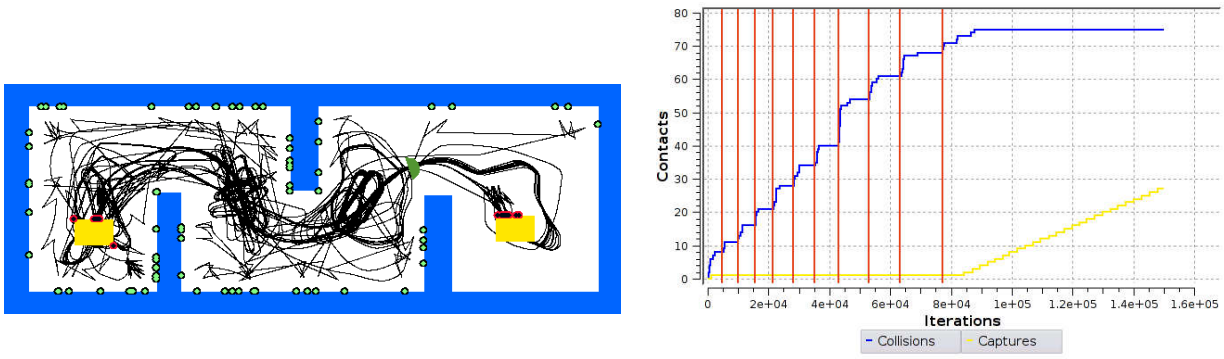
Thirty simulations were run (each with 150.000 iterations). In Fig. 4.1, the best case is shown: no monotony was detected (each monotony is represented by a vertical red line) and 15 repulsive contacts were sufficient for generating suitable navigation behaviors (there are no collisions after 15.000 iterations and exploring behavior that leads to target captures is kept during all the simulation).



a) b)

Figure 4.1: Best case: navigation strategy is learned with few repulsive contacts (collisions); no monotony was detected once the robot keeps exploring the environment and capturing targets (after one target is captured, the other one reappears at the same position); (a) the simulation environment; (b) the correspondent learning evolution graphic.

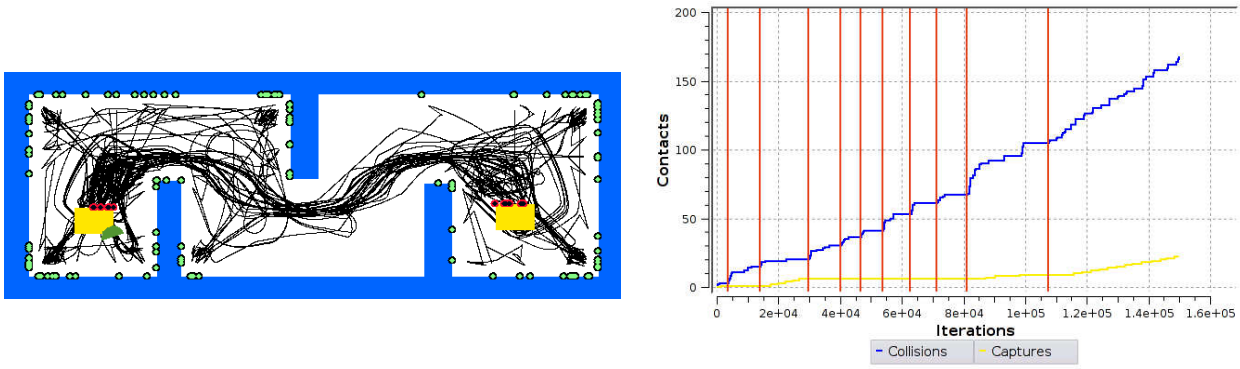
A difficult case is showed in Fig. 4.2: a considerable amount of monotony events are detected (the last one is detected nearly to the eightieth thousandth iteration). After the last monotony event is detected and the correspondent learning by punishment event is fired, the robot develops a suitable trajectory until the end of the simulation (as it can be seen in the learning evolution graphic: the blue line becomes constant and the yellow one keeps increasing with time).



a) b)

Figure 4.2: Difficult case: navigation strategy is learned after several monotony events; (a) the simulation environment; (b) the correspondent learning evolution graphic (detection times for monotony events are represented by vertical lines)

The worst case is shown next (Fig. 4.3): several monotonies events are detected and the number of collisions does not stabilize with time. Although, the number of target captures increases at the end of the simulation. It is possible to note that the environment corners represent a problem in the worst case (Fig. 4.3 (a)): several collisions points (green (clear) circles) can be seen on the corners. Six out of thirty simulations resulted in learning evolution graphics similar to the worst case shown in Fig 4.3 or did not present satisfactory results (with respect to the curves in the learning evolution graphics).



a) b)

Figure 4.3: Worst case: collisions still occur at the end of the simulation (located mainly on environment corners). (a) the simulation environment. (b) the corresponding learning evolution graphic.

Results from 30 simulations are summarized in Table 4.1: mean, standard deviation, maximum and minimum values for number of collisions, number of captures and number of monotonies. Each simulation takes 150.000 iterations. From this table, it is possible to see that 5 to 6 monotonies are detected on average during a simulation experiment. Thus, around 5 learning by punishment events are required so that the autonomous system is able to generate behaviors suitable for constant target captures in this environment configuration (large corridor without obstacles inside).

Table 4.1: Summary for 30 simulations experiments (each composed of 150.000 iterations) - the autonomous system learns by punishment when monotony events are detected.

	No. Collisions	**No. Captures**	**No. Monotonies**
Mean	72,16	47,8	5,66
Standard Deviation	40,14	17,98	3,89
Maximum	168	74	14
Minimum	15	6	0

From above experiments and associated results, the following conclusion is drawn: the negative reinforcement mechanism employed with the monotony detection method do solve the problem of endless cyclic trajectories. The autonomous system does not depend anymore on the type of environment neither on the initial robot position once monotony events are detected and further learning is made possible by the new mechanism based on operant conditioning.

The simulation results are considered good once the autonomous system shows good performance in most of the simulations: their correspondent learning evolution graphics are mostly similar to one shown in Fig. 4.2 (b): the blue curve stabilizes (i.e., there are no more collisions towards the end of the simulation) and the yellow curve goes up (i.e., target captures occur periodically).

5 Stabilization of Number of Collisions

5.1 Introduction

There are situations in which the autonomous navigation system can not avoid collisions with repulsive objects, even after a long training period, but it can still navigate in the environment, capturing attractive objects (possible targets).

For instance, consider the configuration (showed in Fig 5.1) in which the robot detects close repulsive objects on its both sides, while the front sensors of the robot detect a relatively further repulsive object. Depending on the autonomous system experiences (for instance, when there were sufficiently repulsive contacts homogenously over the robot), a collision is very likely to occur on the aforementioned situation. This is possible once the autonomous system can detect similar proximities for both left and right repulsive objects, constituting a risky situation for the robot.

Figure 5.1: Risky configurations for the autonomous navigation system: the robot can go on for one or more inevitable collisions. Left: a corner configuration. Right: a U-shape configuration.

Assuming that the autonomous system learning has not been spatially homogenous (i.e., there were more collisions on one side of the robot than on the other), the situation showed in Fig. 5.1 would be easily overcome by the navigation system. Although, this assumption can imply that the autonomous system is not able to generate efficient trajectories in the environment (e.g., the robot is trapped in an endless cyclic trajectory because it turns very often to the same side – See Chapter 4 *Learning by Punishment*). In this sense, it seems clear that the autonomous navigation system should possess an extra mechanism in order to overcome the eventualities for risky configurations like the one shown in Fig. 5.1.

This chapter describes the proposal of a new repertoire to control the reasoning of PI repertoire neurons, as these neurons are associated with the concept of proximity. Basically, it is expected that these neurons respond differently when the robot faces risky situations. In this way, the navigation system suitably moves the robot away from the risky configuration, influenced by the particular activity of PI neurons.

5.2 Proximity Identifier Reasoning Control Repertoire

The PI repertoire is a bi-dimensional neural network. At each iteration, there is only one neuron (the winner neuron) being activated for each column. Each neuron's output is influenced by two

special parameters that are common for all PI neurons: the acceptance parameter (Θ) and the dispersion parameter (ϕ). The former influences the competition among neurons in a column (described in (17)). The latter influences the slope of the modulator function $f(\phi, n, j)$ defined in (16).

Both parameters Θ and ϕ are fixed (they do not vary with time). However, they seem to be good candidates for influencing PI repertoire reasoning so that navigation decisions are made safer. In this sense, if both parameters can vary with time, there is a way to control PI neurons reasoning.

A new (innate) module, called PI Reasoning Control Repertoire, is added in the system's architecture aiming at the control of PI neurons reasoning (Fig. 5.2). The control is made by adjusting the acceptance parameter $\Theta(n)$ and the dispersion parameter $\phi(n)$. The control repertoire is also connected to every column of neurons in PI neural network.

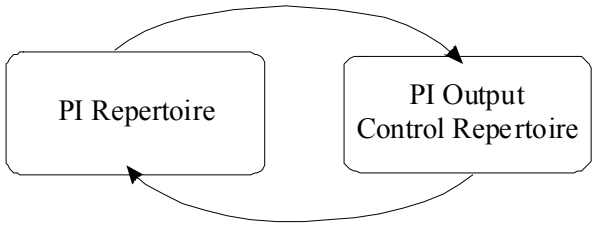


Figure 5.2: Interaction between PI repertoire (a bi-dimensional neural network) and PI Reasoning Control Repertoire (a innate module in the system). The latter influences the output of PI repertoire neurons by adjusting specific parameters of PI neurons.

For each three iterations in PI Repertoire, there is only one iteration in PI Reasoning Control Repertoire. The following adjustment rules (22 to 25) model the PI Reasoning Control Repertoire:

1- Adjustment of the dispersion parameter $\phi(n)$:

$$\phi(n+1) = \begin{cases} \phi(n) + \alpha_1(\tilde{\phi} - \phi(n)) & \text{if } \psi(n) > \overline{\psi}; \\ \phi(n) + \alpha_2(\hat{\phi} - \phi(n)) & \text{if } v(n) > \overline{v}; \\ \phi(n) + \alpha_3(\hat{\phi} - \phi(n)) & \text{otherwise,} \end{cases} \tag{22}$$

where:

$\psi(n)$ is the number of winner neurons in PI neural network that outputs higher than a certain value (at iteration n) (defined in (23));

$v(n)$ is the number of winner neurons in PI neural network that outputs lower than a specific threshold (defined in (24));

$\overline{\psi}$ and $\overline{v}$ are thresholds;

α_1, α_2 and α_3 are update rates for $\phi(n)$;

$\tilde{\phi}$ is a very low value (representing a bottom limit for $\phi(n)$);

$\hat{\phi}$ is a relatively high value lower than 1 (representing a top limit for $\phi(n)$).

$$\psi(n) = \sum_{k=1}^{l} \begin{cases} 1 & \text{if } y_j^k > \bar{y}_a; \\ 0 & \text{otherwise,} \end{cases} \tag{23}$$

$$v(n) = \sum_{k=1}^{l} \begin{cases} 1 & \text{if } y_j^k < \bar{y}_b; \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

where:

j is the winner neuron for column k of PI neural network;

$\bar{y}_a$ and $\bar{y}_b$ are threshold values;

l is the number of columns in PI neural network.

2- Adjustment of the acceptance parameter $\Theta(n)$:

$$\Theta(n+1) = \begin{cases} \tilde{\Theta} & \text{if } \psi(n) > \bar{\psi}; \\ \Theta(n) + \beta_1(\hat{\Theta} - \Theta(n)) & \text{if } v(n) > \bar{v}; \\ \hat{\Theta} & \text{otherwise,} \end{cases} \tag{25}$$

where:

β_1 is an update rate for $\Theta(n)$;

$\tilde{\Theta}$ is a low value (the lowest value for $\Theta(n)$);

$\hat{\Theta}$ is a relatively high value lower than 1 (representing a top limit for $\phi(n)$);

$\psi(n)$, $v(n)$, $\bar{\psi}$, $\bar{v}$ are defined in the same way as in (22).

5.3 Simulation Results

This section shows simulation results that assure the effectiveness of PI Reasoning Control Repertoire on the avoidance of collisions (repulsive contacts) in risky situations. The environment in Fig. 3.8 is chosen for running the next simulations: it contains several repulsive and attractive objects distributed over the inner area. Narrow passages are formed because of the particular configuration of objects in the environment, what makes the navigation more difficult for the robot. These features are well suited for testing the effectiveness of PI Reasoning Control Repertoire in risky situations.

The configuration of the autonomous system parameters are identical to the one presented in Section 4.6 *Simulation Results* from Chapter 4. Concerning configuration of parameters for PI Reasoning Control Repertoire:

$\tilde{\phi}$ =0.007; $\hat{\phi}$=0.4; α_1=0.9; α_2=0.03; α_3=0.4;

$\tilde{\Theta}$=0.2; $\hat{\Theta}$=0.63; β_1=0.08;

$\bar{y}_a$=0.55l; $\bar{y}_b$=0.93l;

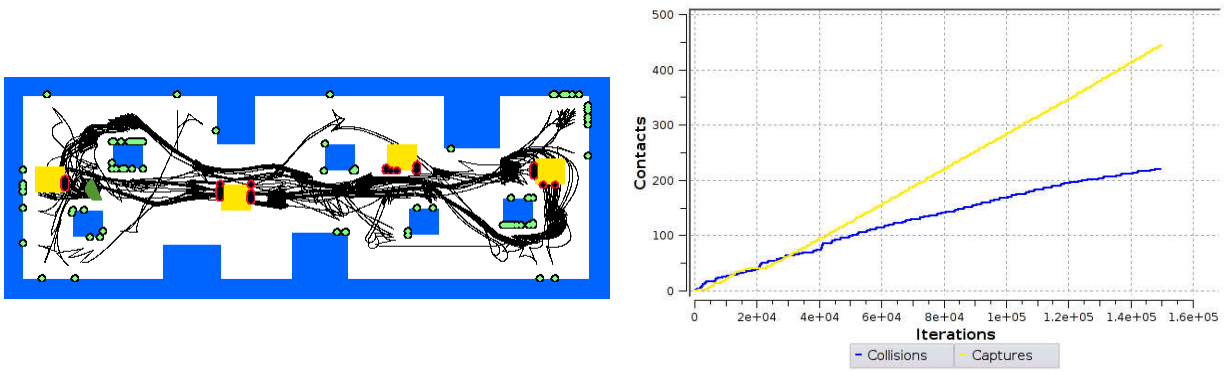
$\Theta(0)=\hat{\Theta}$; $\phi(0)=\hat{\phi}$.

where l is the number of columns in PI repertoire.

Simulation results in which the autonomous system is not influenced by the innate control repertoire are shown first. Following this, tests are made considering that the PI Reasoning Control Repertoire is part of the system architecture. Each simulation lasts 150.000 iterations.

1- Simulations without PI Reasoning Control Repertoire

The best case is shown in Fig. 5.1: the autonomous system learns to exploit the environment (generating repulsive and attractive behaviors to obstacles and targets, respectively); besides, the yellow curve goes up during all the simulation (i.e., target captures are always achieved), while the blue curve also goes up (i.e., no stabilization of collisions is achieved). For this simulation, there were more than 200 collisions, what is not desirable at all.

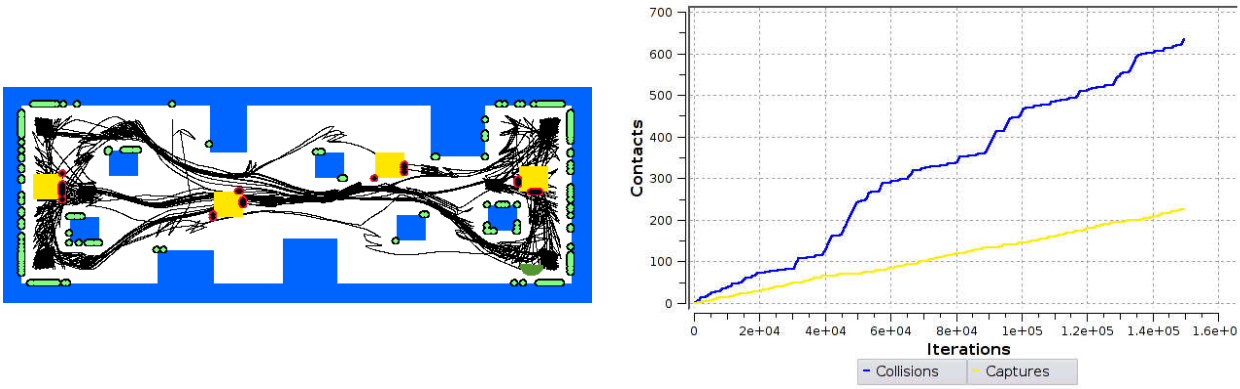


a) b)

Figure 5.1: Best case: foraging behavior is achieved and collisions are still present at the end of simulation. (a) the simulation environment. (b) the corresponding learning evolution graphic (note that the robot captures the target after touching it twice; the yellow curve represents the (accumulated) number of attractive contacts and not the number of target captures).

Results for the worst case are shown in the following figure. From Fig. 5.2 (b), it can be seen that the yellow curve (for attractive contacts) has roughly half of the inclination of the same curve in Fig. 5.1 (b) (the best case), while the inclination of the blue curve (for repulsive contacts) is much bigger than in the best case. The environment representation at the end of the simulation (Fig. 5.2 (a)) shows that the robot was constantly colliding on the corners of the environment

(observe the concentration area of collisions points represented by green (clear) circles), what contributes to the low number of attractive contacts (approximately 220).



a) b)

Figure 5.2: Worst case: foraging behavior is achieved, though the number of collisions is high. (a) the simulation environment. (b) the corresponding learning evolution graphic.

Results for 22 simulations are summarized on the following table. A high mean for the number of repulsive contacts is registered: 445. Although, the mean for the number of attractive contacts is also high (317). As expected, the number of monotonies is very low on average (less than 1) once the particular environment is suited for a diverse learning (i.e., several attractive and repulsive contacts are made possible during the simulation).

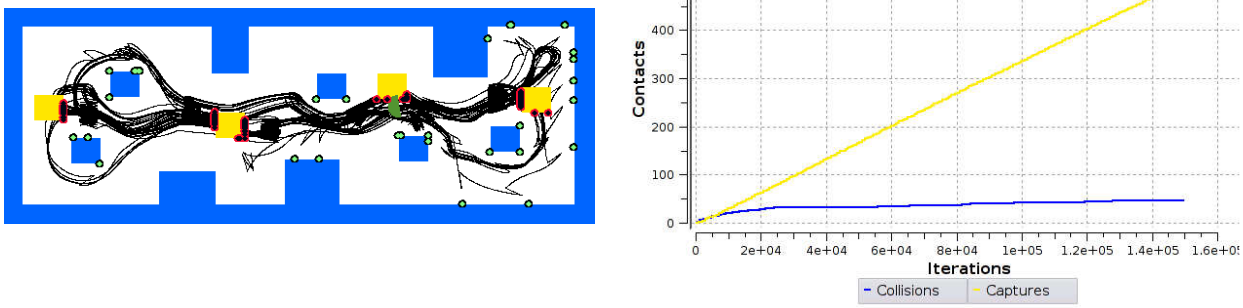
Table 5.1: Summarized results for the first set of simulations (without PI Reasoning Control Repertoire). Number of simulations: 22. Each simulation takes 150.000 iterations.

	No. Collisions	**No. Captures***	**No. Monotonies**
Mean	444.90	317.63	0.36
Standard Deviation	195.17	76.86	0.66
Maximum	750	470	2
Minimum	116	195	0

* *(number of attractive contacts)*

2- Simulations with PI Reasoning Control Repertoire

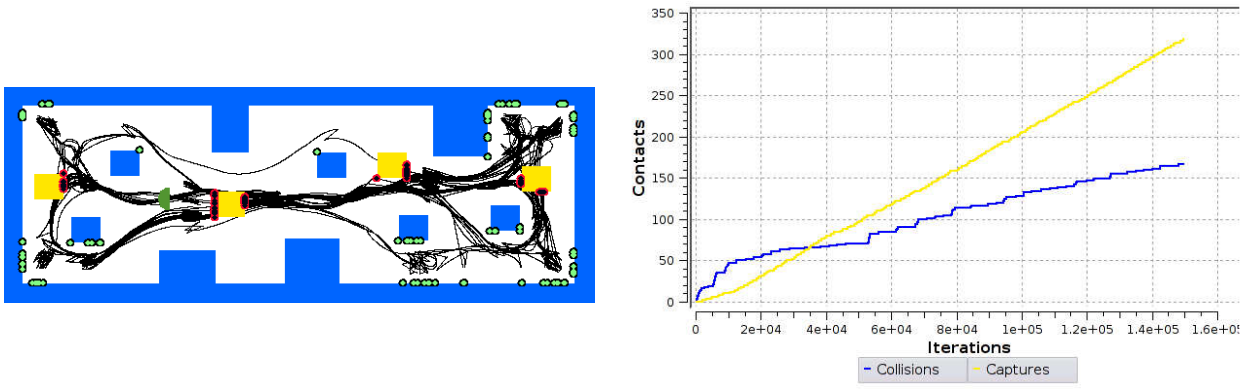
Next figure shows the results for the best case: the autonomous system learns to exploit the environment after few collisions (less than 50): it generates a suitable trajectory until the end of the simulation (nearly free of collisions with obstacles) that leads the robot to a constant number of target captures over time. In Fig. 5.3 (b), it can be seen that the angle between the yellow curve and the blue curve indicates that the achieved autonomous system behavior is very efficient. The best case occurs more often: 15 out of 18 simulations experiments presented similar results.



a) b)

Figure 5.3: Best case: learned exploring behavior is nearly optimal. The robot develops a trajectory nearly free of collisions that leads to constant target captures. (a) the simulation environment. (b) the corresponding learning evolution graphic.

The worst case is shown in Fig 5.4: the autonomous system can not learn a trajectory free of collisions (note that the blue curve keeps going up in the learning evolution graphic - Fig 5.4 (b)); though, target captures are still possible until the end of the simulation. It can be seen that the number of attractive contacts in the worst case (nearly 320) is significantly less than in the best case (512). Three out of eighteen simulations were not satisfactory with respect to the stabilization of number of collisions (indicated by the blue line in the learning evolution graphic).



a) b)

Figure 5.4: Worst case: exploring behavior (foraging) is achieved without stabilization of collisions. The worst case is not common in a simulation with the innate control repertoire (the chance is approximately 15%). (a) the simulation environment. (b) the corresponding learning evolution graphic.

It is possible to observe clearly the effectiveness of PI Reasoning Control Repertoire in risky situations (e.g., narrow passages, environment corners): in 85% of the cases, the stabilization of collisions is achieved. Analyzing the results from the first set of experiments (in which the innate control repertoire is not present in the system architecture), it can be seen that no stabilization of collisions was achieved (e.g., the best case for the first set of experiments (Fig. 5.1) is similar to the rare worst case for the second set of experiments (Fig. 5.4)).

The summarized results are shown in Table 5.2. The mean for the number of collisions (71) is very low compared to the mean of collisions for the first set of experiments, which is 445 (Table 5.1). From this comparison, it is clear that the new innate control repertoire is making a big difference regarding the performance with respect to number of collisions. It is also possible to note that all the values in the column *No. Captures* (Table 5.2) are higher than the respective values in the first case (Table 5.1).

Table 5.2: Summarized results for the second set of simulations (with PI Reasoning Control Repertoire). Number of simulations: 18. Each simulation takes 150.000 iterations.

	No. Collisions	**No. Captures***	**No. Monotonies**
Mean	71.66	384.22	1.28
Standard Deviation	35.57	96.06	2.24
Maximum	167	512	7
Minimum	35	203	0

* *(number of attractive contacts)*

The initial objectives with the introduction of the PI Reasoning Control Repertoire are achieved and simulation results confirm the effectiveness of the new innate repertoire in risky situations (considering a particular environment). With the addition of this new repertoire, the (mean) number of repulsive contacts (collisions with obstacles) are substantially decreased while the (mean) number of attractive contacts are slightly increased. Thus, autonomous systems endowed with the new control repertoire are more adaptable to the environment: pleasurable (attractive) contacts are kept while aversive experiences (collisions with obstacles) are reduced to nearly zero after a initial learning phase.

6 Further Analysis of the Autonomous Navigation System

6.1 Introduction

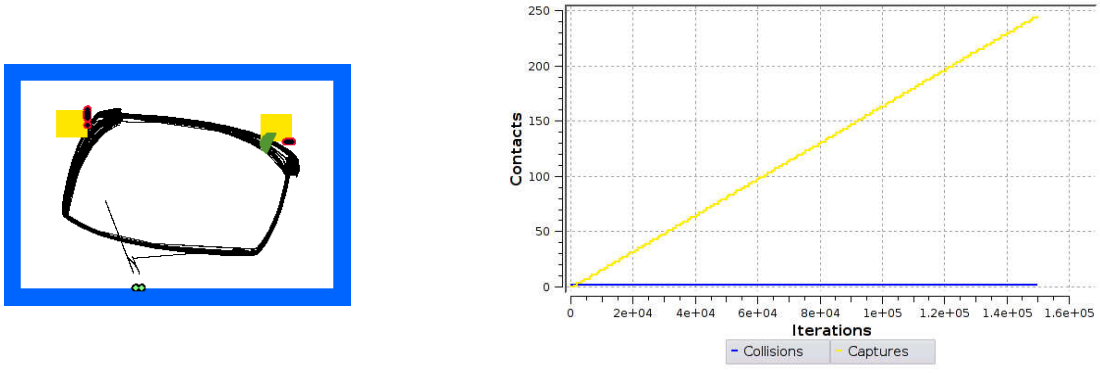
This chapter shows several simulation experiments and associated results considering that the autonomous system is able to detect monotonies (and learn by punishment, see Chapter 4) and that the PI Reasoning Control Repertoire is present in its architecture (Chapter 5); otherwise, a particular configuration is mentioned. The system's parameters are configured in the same manner as in last chapter. Consider also the explanations relating to simulations in Section 3.6 *Configuration of Simulation Experiments* in Chapter 3.

Simulations are executed in simple and complex (maze) environments. Dynamic environments (where objects have simple moving behaviors) are also considered. Furthermore, generalization capabilities of the autonomous system are tested: a simulation takes place in a training environment and subsequently in a test environment. Experiments considering multiple robots in a same environment are accomplished as well.

6.2 A Simple Experiment

The following experiment is accomplished in a simple environment without obstacles inside and with two attractive objects. Fifteen simulations are run (each of 150.000 iterations). No previous training is done (i.e., the autonomous system does not have any previous acquired-knowledge or navigation behaviors). As the environment is rather simple, the PI Reasoning Control Repertoire is not considered in the system's architecture for the simulations in this section.

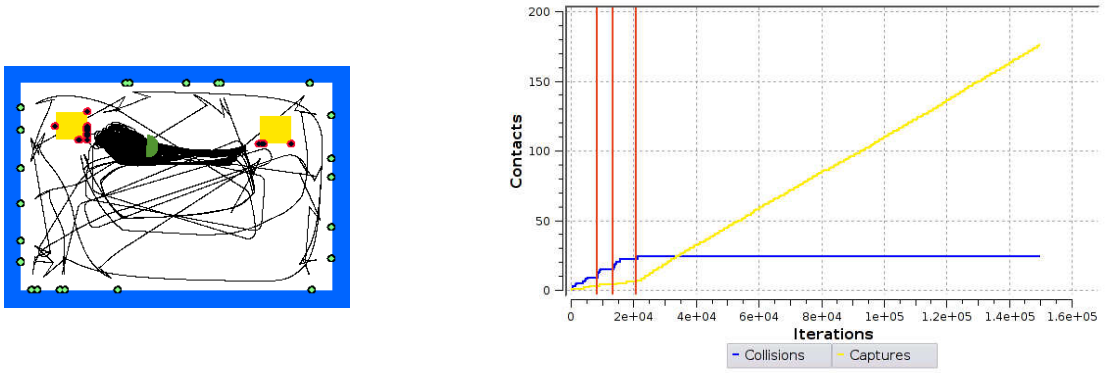
The best case can be seen in Fig. 6.1: attractive and repulsive behaviors are achieved with only 2 collisions. The robot develops rectangular-shaped trajectories in most of the simulations (with distinct inclinations). As the mean number of collisions is very low (Table 6.1), it is possible to conclude that the robot achieves suitable behaviors for the particular environment using only a small part of the neural architecture (i.e., only few neurons are activated).



a) b)

Figure 6.1: Best case: behavior suitable for target captures is achieved after 2 collisions with obstacles. (a) the simulation environment. (b) the corresponding learning evolution graphic.

Next figure shows results for the worst case: 24 repulsive contacts and 3 monotonies were registered during the simulation. Despite the simulation numbers are the worst, the final robot trajectory is unique in this simulation: its shape is restricted to a wide line (instead of a rectangle).



a) b)

Figure 6.2: Worst case: 24 collisions were registered in the beginning of the simulation and final robot trajectory has a particular shape. (a) the simulation environment. (b) the corresponding learning evolution graphic.

The results for 15 simulations are summarized on the next table. The maximum number of collisions is 24 while the maximum number of target captures is 265. These results are not surprising because of the simplicity of the environment.

Table 6.1: Summarized results for the simulations with the simple environment. Number of simulations: 15. Each simulation takes 150.000 iterations.

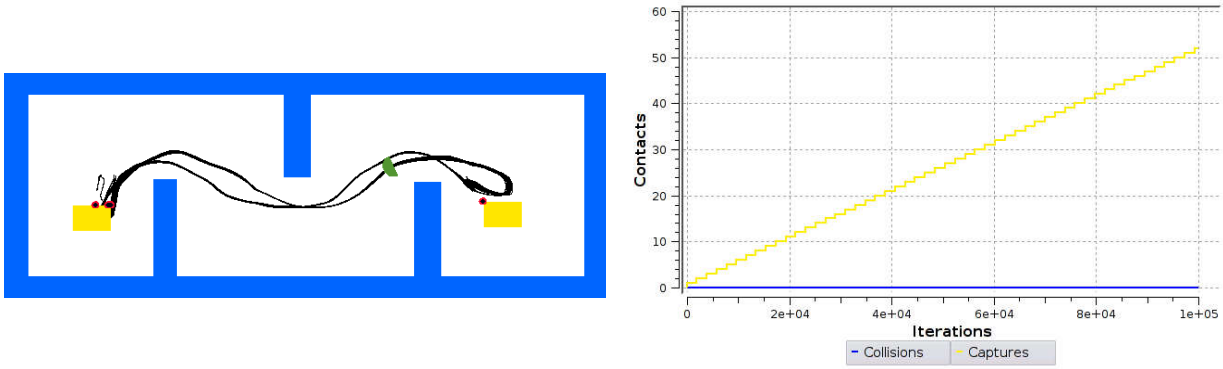
	No. Collisions	**No. Captures**	**No. Monotonies**
Mean	8.33	166.8	0.73
Standard Deviation	5.66	48.52	0.8
Maximum	24	265	3
Minimum	2	106	0

6.3 Generalization Capabilities

This section shows simulation experiments aiming at testing the generalization capabilities of the proposed controller. For this purpose, first the robot is inserted in a training environment (shown in Fig. 3.8, consider also the two different initial robot positions in that figure): there are several obstacles and targets in the environment (creating distinct and narrow passages). After 150.000 iterations in the training environment, the robot is inserted in the test environment shown in Fig 6.3 (a large corridor with a target located at each extremity).

The best case is shown in Fig. 6.3. The autonomous system efficiently guides the robot in the corridor such that no collisions occur and target captures are periodic until the end of the simulation (see the learning evolution graphic in Fig. 6.3 (b)). Five out of 18 simulations were nearly identical to this case: no collisions takes place and number of target captures is higher than 40; in 12 experiments (67%), the learning evolution graphic shows a desired shape: the blue line

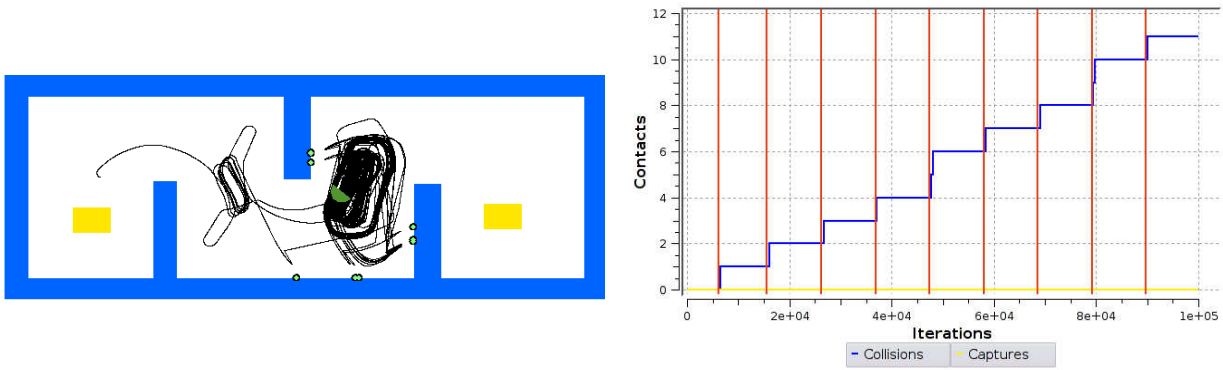
(accumulated number of collisions) ends constant (or stays constant for a long period) and the yellow line keeps increasing most of the time during the experiment (targets are being captured).



a) b)

Figure 6.3: Best case: good generalization of acquired knowledge: periodic target captures take place without any collisions. Frequency: common. (a) the simulation environment. (b) the corresponding learning evolution graphic.

The worst case can be seen in the next figure. The autonomous system generates a cyclic trajectory during all simulation. Several monotonies events are detected; though no suitable behaviour is achieved. This case has occurred only once. The training phase for this case probably was problematic. No generalization is achieved. In 6 out of 18 experiments the learning evolution graphic shows that the generalization is not achieved (number of collisions is still a problem).



a) b)

Figure 6.4: Worst case: problematic simulation without target captures and with several monotonies events. .Frequency: rare (1 case detected) (a) the simulation environment. (b) the corresponding learning evolution graphic.

The summary is show in the following table. The mean value and the maximum value for the number of captures are 35 and 52 respectively whereas the mean number of collisions is relatively low: 28. The mean number of monotonies is very low: 3.

Table 6.2: Summarized results for the generalization capabilities tests. Number of simulations: 18. Each simulation takes 150.000 iterations.

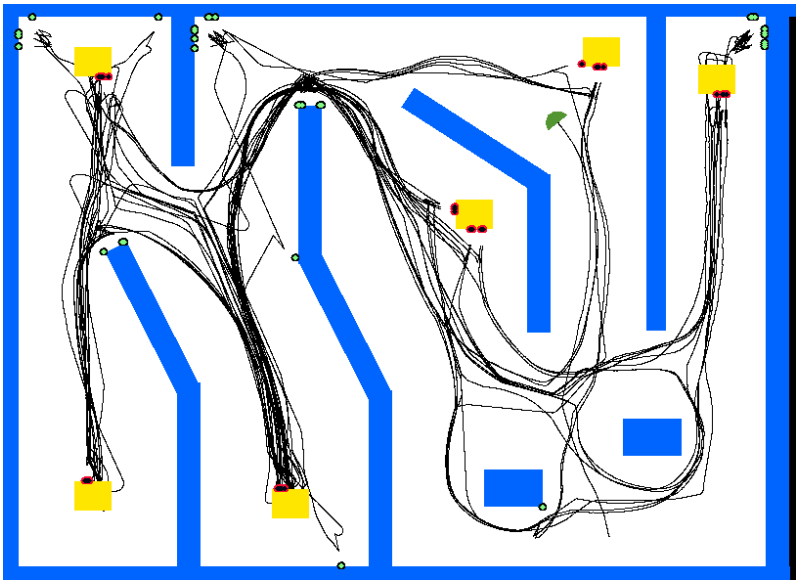
	No. Collisions	**No. Captures**	**No. Monotonies**
Mean	28.39	35.5	3.39
Standard Deviation	33.98	14.34	3.4
Maximum	120	52	9
Minimum	0	0	0

6.4 A Complex Environment

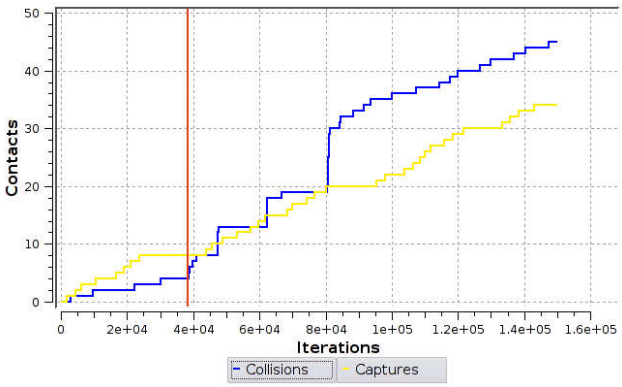
Training and test environments are considered for the following simulations. The training environment is shown in Fig. 3.8 (consider two different initial robot positions as well, shown in such figure). The summary of the training phase (which lasts 150.000 iterations for each simulation) is shown in Table 5.2.

The test phase takes place in the environment shown in the next figure (Fig. 6.5 (a)): the maze environment. The environment is large and composed of particular places and passages; in addition, (six) targets are distant located from each other (remember that targets reappear at the same position after the last target is captured – see Section 3.6 *Configuration of Simulation Experiments* in Chapter 3). In the test phase, the initial robot position is fixed for all simulations. Furthermore, the parameter time interval for the monotony detection is set to $\lambda(0) = 8000$ (a monotony event should take longer time to be detected in the test phase).

The best case is shown on the next figure: the robot develops a rather efficient trajectory in the maze, being capable of capturing all targets. In this particular simulation, the autonomous system is very efficient because the robot captures each target at least 5 times. One monotony is detected during the simulation; and collisions (with obstacles), concentrated on environment corners, occur sporadically during the simulation.



a)



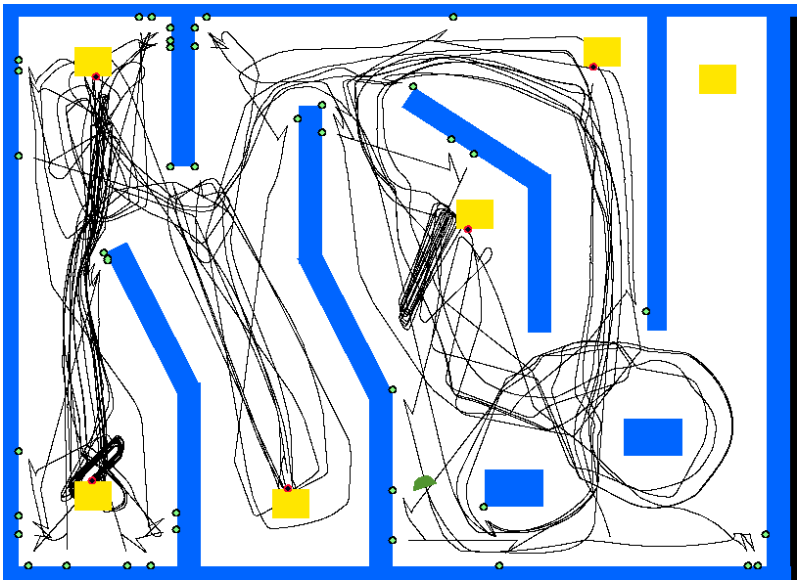
b)

Figure 6.5: Best case: robot captures every target at least 5 times. (a) the simulation environment. (b) the corresponding learning evolution graphic.

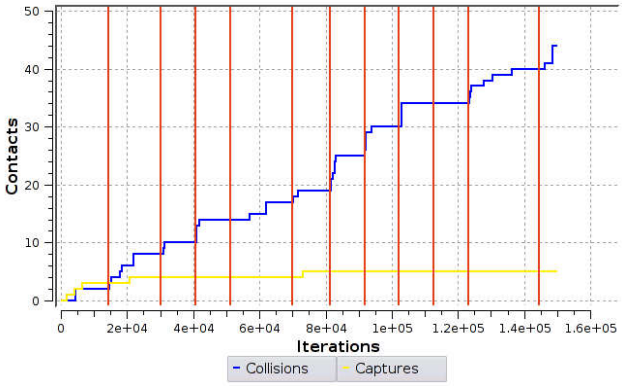
The following figure shows the worst case: the autonomous system is not able to guide the robot to the upper-right located target. Once all targets must be captured in order to reappear in the environment, the number of attractive contacts (that is, captures) is limited to 5 (see Fig. 6.6 (b)). Eleven monotony events are detected occasionally during the simulation. Some specific areas in Fig 6.6 (a) indicate clearly monotony situations (by checking the robot trajectory): the central area, the bottom left area and possibly around two objects on the bottom right area.

The results are summarized in Table 6.3. The robot captures 13 targets on average in a simulation. This is to say that all targets in the environment are captured at least twice, on average. It is possible to note that the number of collisions in the best case above (that is 45) is between the mean number of collisions (70) and the minimum number of collisions (19). Very low number of collisions for this environment is usually associated with high number of monotonies.

In 3 out of 18 simulations, the robot did not capture one target. There were 2 cases in which the upper-right target was not captured and 1 case in which the central target was not captured.



a)



b)

Figure 6.6: Worst case: robot does not capture the upper right target (probably the most difficult located target). (a) the simulation environment. (b) the corresponding learning evolution graphic.

Table 6.3: Summarized results for the simulations with the maze environment (and improved controller). Number of simulations: 18. Each simulation takes 150.000 iterations.

	No. Collisions	**No. Captures**	**No. Monotonies**
Mean	70.28	13.44	6.39
Standard Deviation	28.17	7.49	3.15
Maximum	122	34	13
Minimum	19	5	1

Experiments with the original controller [28] were also accomplished in this environment in order to make possible comparisons with the extended (improved) autonomous system. A summary is shown in Table 6.4. Ten simulations (of 150.000 iterations each) were run (consider the same training phase conditions as explained in the beginning of this section). In only one experiment the number of target captures reached 14 (and with no collisions): best performance.

A high number of collisions take place when the robot gets trapped in a corner configuration (innate reflexes of the original autonomous system are not appropriate enough to make the robot escapes from the corner configuration). This case was detected in 4 out of 10 simulations.

In 5 simulations the autonomous system did not generate suitable exploring behaviors (excluding the cases when the robot gets trapped in corner configurations): the robot develops cyclic trajectories for long periods, what has a negative impact on the number of target captures.

Table 6.4: Summarized results for the simulations with the maze environment (and original controller). Number of simulations: 10. Each simulation takes 150.000 iterations.

	No. Collisions	**No. Captures**
Mean	1221.6	3.8
Standard Deviation	1548.22	3.79
Maximum	3077	13
Minimum	0	1

Simulations results show clearly the advantages of the proposed controller over the original controller [28]. With the new improvements, cyclic trajectories are detected and further environment exploring (with target captures) is made possible.

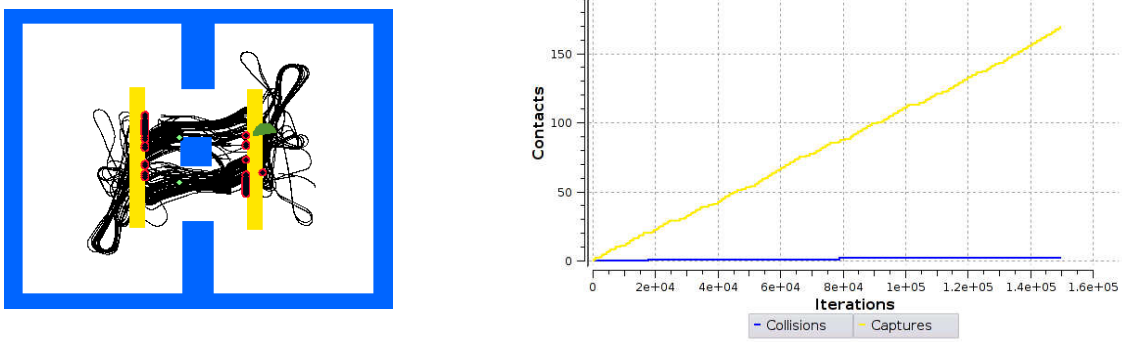
6.5 Dynamic Environments

There are two stages for each simulation: the training phase and the test phase. The training phase takes place in the environment shown in Fig. 3.8. The summary of the training phase is presented on Table 5.2. Each simulation phase takes 150.000 iterations.

The test environment is composed of two repulsive compartments next to each other (Fig. 6.3 (a)). Such compartments are connected through a passage partially blocked by a moving obstacle. The moving obstacle is initially at the center of the passage. Once the simulation starts, the special obstacle moves on the vertical direction with half of the robot velocity (i.e. 0.14 distance units). The moving obstacle is limited up and down by other obstacles (a collision with another object generates a change on the movement orientation). Additionally, a long attractive object is placed in each compartment in order to force the robot into the risky passage (a passage with a moving obstacle). Furthermore, the parameter time interval for the monotony detection is set to $\lambda(0) = 8000$ in the test phase (a monotony event should take longer time to be detected in the test phase).

The best case is shown on the following figure. Only two collisions (with the moving obstacle) are registered during all the simulation. The robot develops a very safe trajectory (Fig. 6.7 (a)) and the targets are periodically captured (see the yellow (clear) curve on the learning evolution graphic – Fig. 6.7 (b)). In 84% of the simulations, the corresponding learning evolution graphics show a constant blue line and a increasing yellow line at some point in the simulation for a big

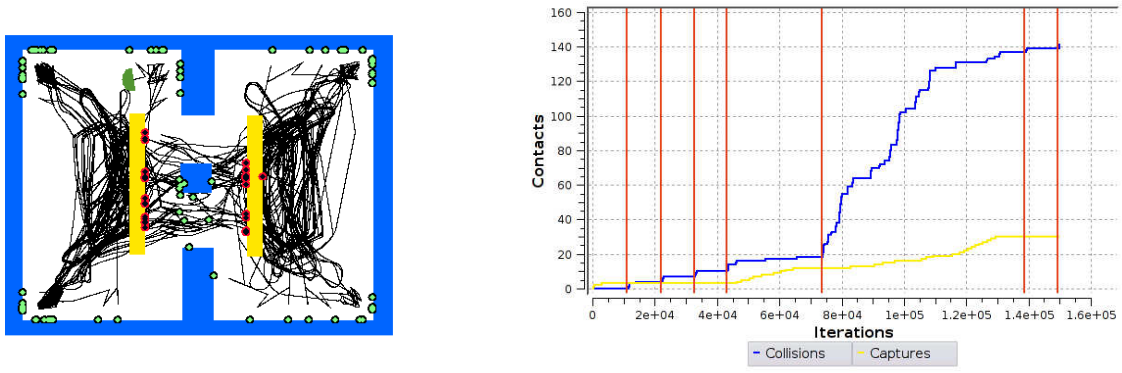
time interval (i.e., the angle formed between the yellow curve and the blue curve is rather similar to the angle in Fig. 6.7 (b)).



a) b)

Figure 6.7: Best case: autonomous system provides the robot with a safe trajectory. Targets are occasionally captured. Similar cases are frequent. (a) the simulation environment. (b) the corresponding learning evolution graphic.

The following figure shows the worst case: the robot trajectory is not safe (several collisions occur); seven monotonies situations are detected and the number of target captures is the lowest among all simulations. The environment corner is a problem in this case, what contributes significantly to the high number of collisions. Similar cases are detected in 16% of the simulations.



a) b)

Figure 6.8: Worst case: collisions occur until the end of the simulation and 7 monotonies are detected. Similar cases are not frequent. (a) the simulation environment. (b) the corresponding learning evolution graphic.

The following table shows the simulation results in terms of mean, standard deviation, maximum and minimum values. The minimum number of collisions is 1 while the mean for the number of captures is 90. The maximum number of captures is 169.

Table 6.5: Summarized results for the simulations with the dynamic environment. Number of simulations: 18. Each simulation takes 150.000 iterations.

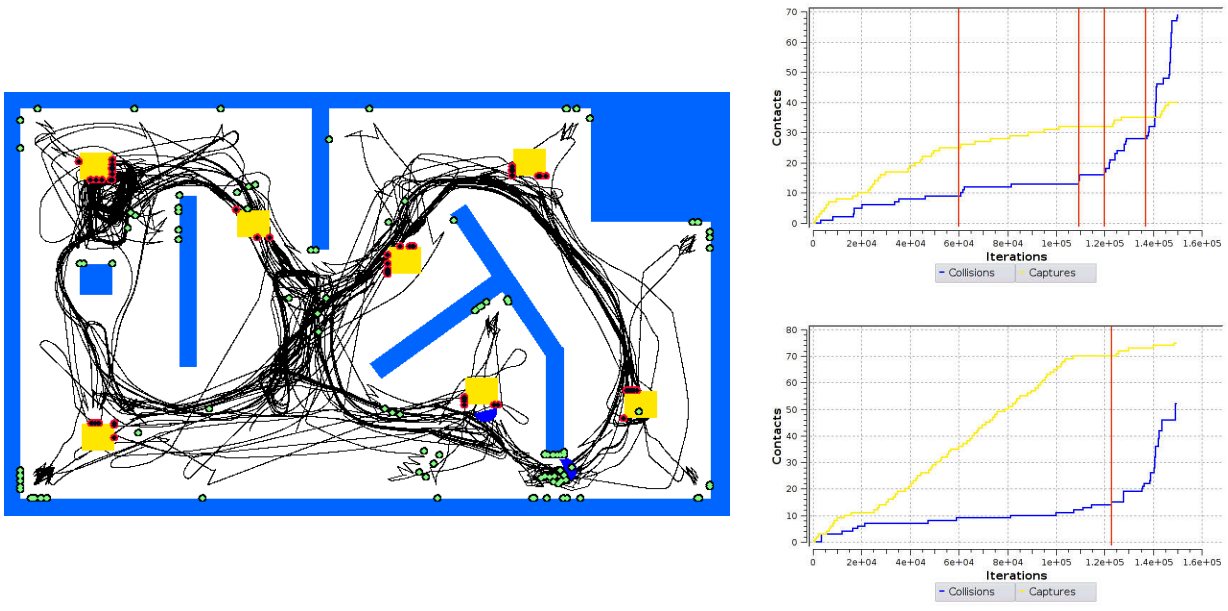
	No. Collisions	No. Captures	No. Monotonies
Mean	39.28	89.94	0.67
Standard Deviation	39.31	35.02	1.68
Maximum	142	169	7
Minimum	1	30	0

Simulation results show that the autonomous system is adaptable to the particular dynamic environment: in 84% of the cases, the robot does not collide with obstacles for a big time interval (mainly towards the end of the simulation) while it captures targets periodically. On average, the number of target captures is high and the number of collisions is low.

6.6 Multiple robots in an environment

Interesting experiments can be accomplished using the simulator for autonomous navigation. For instance, multiple robots can be placed in the same environment. In this sense, interesting conclusions can be drawn.

Consider the simulation environment shown in Fig. 6.9. There are 7 targets located in special positions in the environment. First, 2 simulations (each of 150 thousand iterations) are executed considering 2 robots in this environment (Fig. 6.9). Before placing the robots together, each robot is trained in a previous environment (shown in Fig. 3.8) for 150.000 iterations.



a) b)

Figure 6.9: Two robots experiment: 2 robots share the same environment, exploring it and capturing targets. (a) the simulation environment. (b) the learning evolution graphic for both robots.

The trajectories in Fig. 6.9 - (b) (black lines) indicate that the robots explore the most important parts of the environment several times. In addition, it can be seen (Fig. 6.9 - (b)) that one robot captures more targets than the other robot. If the situation is considered of competitive type, the winner robot (the one that survives) would be the one that captures more targets and collides less with obstacles (corresponding to the second learning evolution graphic).

On the other hand, the robots could be working together in the same task of capturing targets (cooperative robots). In this sense, two robots capture more targets working together than working by themselves. Table 6.6 shows the number of collisions with obstacles and number of target captures for two simulations. In the first simulation, total number of collisions (for both robots) is 121 while the target captures sums up to 115. Most of the collisions are between robots and with the corners of obstacles.

Table 6.6: Total number of collisions and total number of captures for the case in which 2 robots are in the same environment.

No. Collisions	No. Captures
121	115
67	94

Second, 4 (already trained) robots are placed in the same environment and 2 simulations are run (Fig. 6.10). Results from a simulation are shown in the next figure. It can be seen that the trajectories cover a very greater area than in the case of 2 robots above. There is no robot with superior performance than the other robots (i.e., every learning evolution graphic is similar to each other to some extent – Fig. 6.10 (b) (c) (d) (e)). Thus, a competitive environment would not come up.

On the other hand, the result from a (implicit) cooperation (between robots) can be easily verified. On average, the number of target captures is slightly higher than the double number of captures in the case of 2 robots. Thus, as the number of robots is multiplied by two, the number of target captures is also duplicated, plus a small factor. There are some reasons for the appearance of this small factor. For instance, when there are more robots in the same environment, each robot needs to navigate possibly a less distance between each target capture; difficult located targets have a higher probability of being captured (thus, causing the replacement of all other targets in the environment); and monotonies events are less probable once more interactions between robots are made possible.

The following table shows the results for 2 simulations. Despite the increase in the number of target captures, the number of collisions is also multiplied by the same factor. This is because whereas the number of robots is higher the size of the environment is not made larger, what causes constant collisions between robots.

Table 6.7: Total number of collisions and total number of captures for the case in which 4 robots are in the same environment..

No. Collisions	No. Captures
257	246
220	280

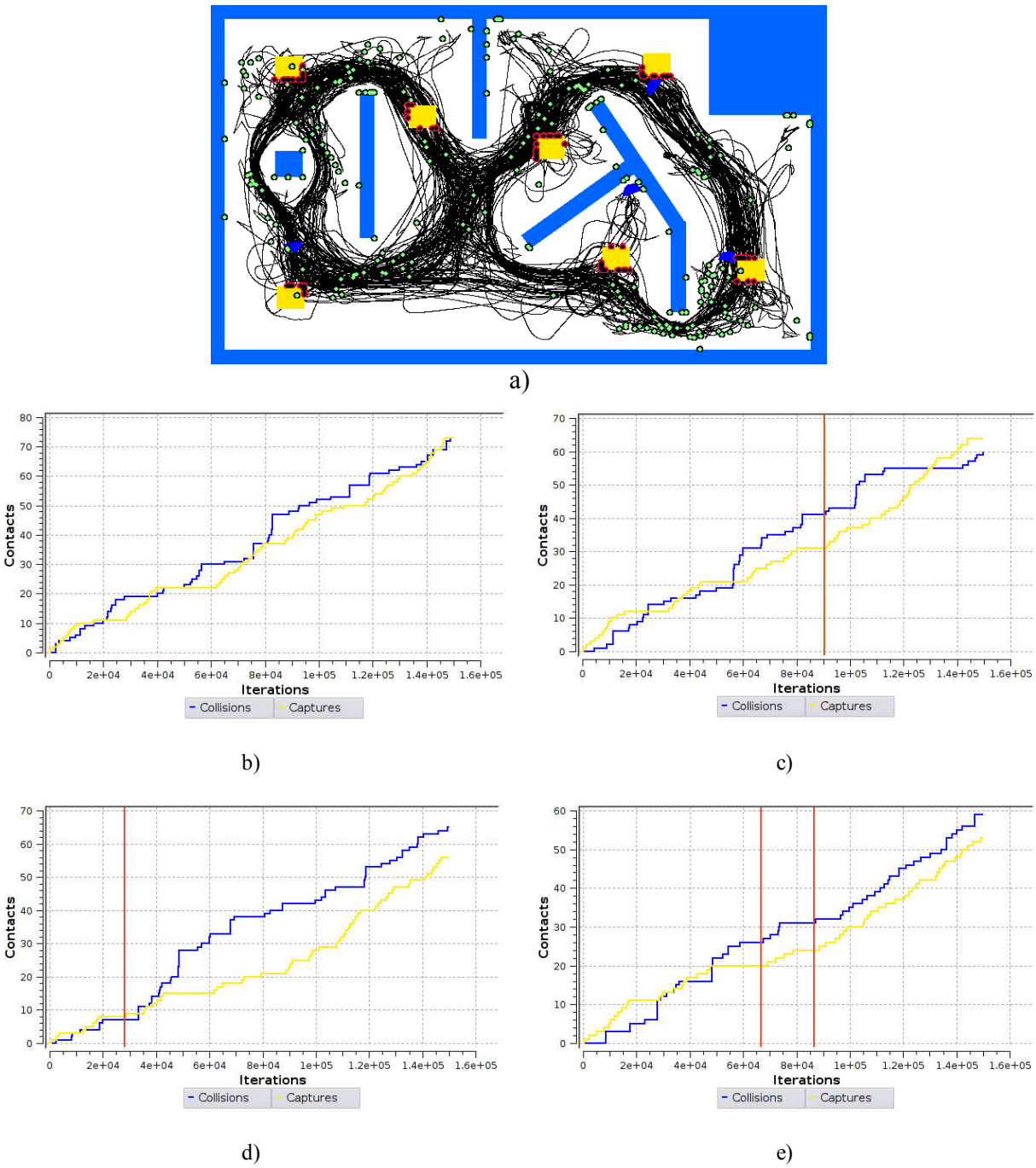


Figure 6.10: Four robots experiment: 4 robots share the same environment, exploring it and capturing targets (a) the simulation environment. (b) the learning evolution graphic for both robots.

Simulation results show that multiple robots exploring the same environment contributes to the improvement of performance with respect to the total number of target captures. Furthermore, individual robot performance (with respect to target captures) is also improved when multiple robots work together (cooperation).

7 Conclusion

7.1 Introduction

This work proposes additional improvements on a particular autonomous navigation system described in [28] and carries out new experiments in order to show more accurately the particularities and potentialities of the system.

The environment model is composed of different classes of objects, each of which associated with a respective color. The robot sensors, to detect the colors of objects and the distance from (the closest) objects to the robot, are not specific either for targets (attractive objects) or for obstacles (repulsive objects). Contact sensors provide positional contact information and innate repertoires in the system detect the nature of the interaction - i.e., repulsive (aversive) or attractive (pleasurable) contact. The autonomous system in [28] is composed of a hierarchical and modular neural network with particular neurons reasoning and connection arrangements. Through interaction with the environment, the autonomous system learns to distinguish different classes of objects (generating specific behaviors for each one of the classes). The system also shows generalization capabilities for distinct environments. Besides it learns to take navigation decisions to move the robot to the closest attractive object (target).

It is possible to observe certain deficiencies with the former autonomous system. Firstly, there is dependence between initial simulation conditions (e.g., position of the robot, environment configuration) and the performance of the system with respect to the capacity of exploring the environment). In other words, the robot (guided by the autonomous system) can be trapped in an unsuitable cyclic trajectory unless some initial simulation configuration is carefully chosen so that sufficient interactions with the environment are made possible (for the system generates suitable exploring behaviors).

Risky situations for the robot (e.g., corner configuration and U-shaped configuration, see Fig 5.1) are often associated with collisions even after learning. The original autonomous system is not always able to escape such configurations, what can debilitate the performance with respect to number of collisions.

The proposals of new artificial learning and control mechanisms for the enhancement of the autonomous system in [28] are described in Chapter 4 *Learning by Punishment* and Chapter 5 *Stabilization of Number of Collisions*. The former chapter presents the first changes in the architecture and additional learning mechanisms aiming at diminishing the problem of unsuitable cyclic trajectories. By detecting monotony events (that are associated with cyclic trajectories), learning events are followed in order to decrease the probability that the autonomous system shows the inappropriate behavior in the future. This learning mechanism is based on operant conditioning [29]. Simulation results confirm the effectiveness of the improved autonomous system: the detection of monotonies events makes possible further learning and consequently the exploring behavior (for capturing targets, foraging) is gradually refined.

The latter chapter describes a new innate repertoire that controls specific parameters from Proximity Identifier (PI) Repertoire: the PI Reasoning Control Repertoire. This control repertoire can be considered as an inborn mechanism acquired through an evolutionary processes in order to generate individuals more adaptable to the environment (i.e., individuals that can avoid risky situations and suffer a less number of aversive interactions). Although the evolutionary process is not modeled in this work: the control repertoire is defined by a set of devised equations. The control rules influence the activation of neurons in the PI repertoire. If the current situation is

risky (this is detected by the control repertoire), then a differentiated activation of PI neurons follows the influence of the control repertoire in order to lead the robot away from the risky configuration. Simulation results ratify the performance of the improved autonomous navigation system (with the new innate repertoire): the number of collisions is dramatically reduced for a particular diverse environment (with several targets and obstacles creating narrow passages and risky configurations).

Chapter 6 presents several experiments for analyzing system potentialities in different environment configurations. Tests with a simple environment resulted that with very few collisions the autonomous system learned to navigate in the environment and capture the targets. Generalization capabilities are verified through training the autonomous system in a first environment and testing it in a second (and distinct) environment: in several cases the robot did not collide at all (in the test phase). A maze environment (containing 5 targets) is also used to test the autonomous controller: in the best case, the autonomous system captures at least 5 times each target for a simulation of 150.000 iterations (despite it does not have a declarative memory system for storing information about targets locations). Experiments with a dynamic environment are also done, showing that in several cases, the autonomous system was able to adapt itself to the environment dynamics (reducing the number of collisions with a moving obstacle and increasing the number of target captures). Finally, experiments with multiple (previously trained) robots are accomplished: as the number of robots is multiplied by 2 the total number of target captures is also doubled (or more than doubled). It can be seen that the individual robot performance (with respect to number of target captures) is also improved: the cooperation in the task of seeking targets influences positively each autonomous system.

7.2 Discussion and Future work

Simulation results have shown not only the virtues of the proposed navigation system but also some associated deficiencies.

It is possible to observe that even with the new improvements there are still worst cases during the simulations experiments. For instance, in one simulation (for testing the stabilization of collisions) the risky situations are not so efficiently avoided. Even considering that this result is not frequent, it is not sound to affirm that the autonomous controller is optimal (or efficient) for a given task. In fact, this randomness of results resembles what happens with biological systems. Considering that two individuals with the same genetic code (the same autonomous system architecture) have distinct environmental experiences, it is rather possible to foresee that each individual will show distinct behaviors given particular sensory cues (e.g., twins acquire distinct skills during life). This analogy (and the randomness present in the connection arrangements between neurons) would explain the distinctness of generated (learned) behaviors for the proposed autonomous system.

Better results with the mechanism of stabilization of number of collisions can be achieved if an evolutionary process is used instead. This process is devised next. The evolutionary process would be fired when a risky situation is detected by some innate mechanism. The population of individuals corresponds to possible adjustments on the two specific parameters of PI neurons (the same parameters considered before): acceptance parameter (that influences the competition between neurons) and the dispersion parameter (that influences the intensity of neuron output). In this sense, each individual chromosome is composed of a set of pairs of parameters, each pair being associated with a column of neurons in PI neural network. The first generation is randomly

created. The best individuals (selected for reproduction) are those that cause the strongest adjustments on the direction of the robot (this can be measured by the output of the actuator neuron). After sufficient learning processes (that is, after several generations), the steady population would have the individuals most suitable for generating adjustments on the parameters of PI neurons when risky situations are detected.

It is also possible to note that the mechanism of negative reinforcement (used to decrease the probability that the autonomous system shows the unsuitable cyclic behavior in the future) requires a lot of time to be effective, in some simulations. This can be checked by the number of monotonies detected during a simulation. However, this slowness in the process is due to the particular environmental interactions that do not provide the autonomous system with appropriate learning (for instance, contacts are often located over a particular area in the sensorial field, and consequently only few neurons are activated). One could think about a symmetry learning mechanism that fires learning events on both parts (left and right) of the neural architecture when a contact is detected. Apparently this could solve the problem of cyclic trajectories and speed up the learning. Though, there are no evidences that this learning feature is present in biological systems. Furthermore, this symmetry mechanism is not sound once contacts on the left part of the robot generate reflexes opposite to the reflexes generated by contacts on its right part. This probably would enhance the probability that the system presents cyclic trajectories. Thus, a reactive system (like the proposed autonomous system) can not have this symmetry feature. Even if this feature was sound and assuming that two opposite reflexes were generated for a single stimulus, the importance of risky situations would be increased and collisions would be roughly inevitable (once the left part of the architecture would be almost identical to its right part; a corner configuration would stimulate equally both parts of the architecture and the adjustment on the robot direction would be null). On the other hand, some spatial relationship betweens neurons in neighbor columns could be exploited in order to speed up learning.

Appendix A – Neural Networks

The brain is a highly complex, nonlinear and parallel information processing system. Its main constituent cells, known as neurons, are organized in such a way that certain tasks (e.g., pattern recognition, perception and motor control) are computed much faster than in the fastest existent digital computer.

It is expected that there are about 10^{11} neurons in the brain. Neurons collect input signals through structures called dendrites. The input signals are processed in the soma and the resulting output signals are transmitted to other neurons through axons and its ramifications (Figure A.1). The area between an axon of a neuron and a dendrite of another neuron is called synapse. It plays an important role in the transmission of signals between cells and consequently in the brain characteristics as a whole. A synapse can alter a signal originating from an axon or it can even obstruct the signal transit to a dendrite. Furthermore, synapses can be modified during signal transit between neurons, what make probable their association with memory and learning [15].

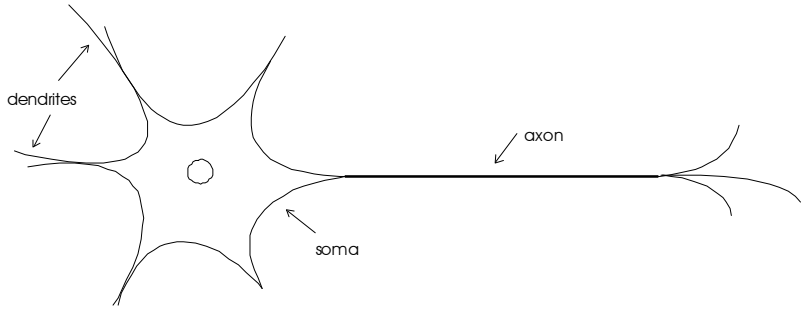


Figure A.1: Biological neuron model.

McCulloch and Pitts proposed the first mathematic neuron model in 1943 [30]. In their work, they describe the mathematics of neural networks. The proposed model is based on the neurophysiologic plausibility of a neuron with a soma function and a threshold, where the weights represent the biological neuron synapses. The researchers showed that a neural network with a sufficient number of neurons and with a suitable learning procedure could approximate any computable function [15].

Figure A.2 shows the formal model proposed by McCulloch and Pitts, where $x = [x_1, x_2, ..., x_n]^T$ is the input vector and $w = [w_1, w_2, ..., w_n]^T$ is the synaptic weights vector related to the neuron memory. The neuron output is calculated by:

$$y = f\left(\sum_{i=1}^{n} x_i w_i\right)$$

where: f is a nonlinear threshold function (the output is high when the sum exceeds a certain limit; otherwise, the output is very low).

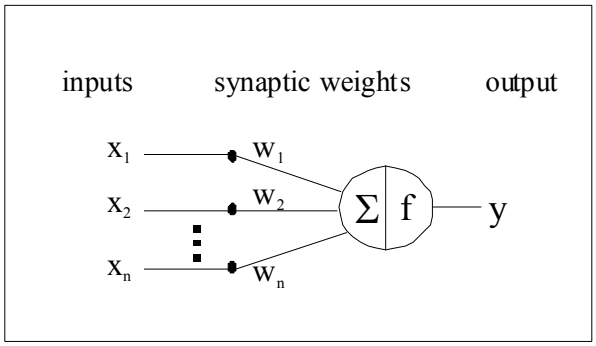


Figure A.2: Formal neuron model.

There are a great number of artificial neural networks models in the literature. In general, these models have one of the following architectures (or a combination of them): one-layer architecture, multi-layer architecture or recurrent architecture (Figure A.3).

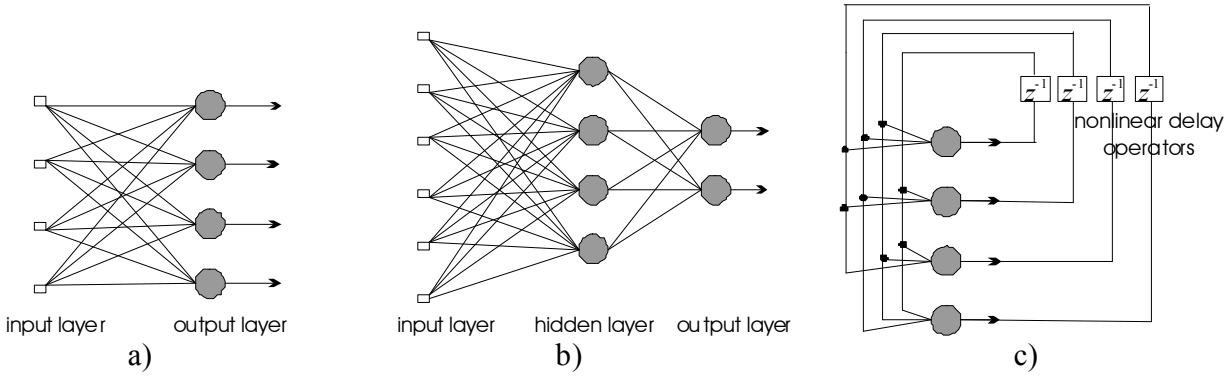


Figure A.3: Neural networks architectures: a) one-layer; b) multi-layer; c) recurrent.

Besides the diversity of architectures for neural networks, there is also a variety of learning algorithms for the adjustment of network structure. One of the methods more disseminated for the learning of multi-layer networks is the gradient-descent method [19]. In general, the adjustment calculation (known as backpropagation) is executed in the opposite direction to the signal propagation in the network. Additionally, this method is supervised once it is necessary to know the output for each input during the learning phase.

Competitive methods provide bases for the proposal presented in [16], where neurophysiologic observations are reproduced. These methods are applied in neural networks whose neurons develop a competitive dynamic among themselves, i.e., for each input to the neural network only the winner neuron and its neighborhood neurons remain active after the neural network reaches the equilibrium state [15]. The synaptic weights adjustment depends on the final competition result.

Self-organizing maps (SOMs) [17] emulate biological neuronal systems. They are capable of forming topological maps on their architecture according to the topological characteristics existent in the input signals. The learning method is non-supervised and competitive. The winner neuron is determined according to the distance between input vector and synaptic weights vector.

Appendix B – Simulator for Autonomous Robot Navigation

B.1 Introduction

A special computational environment was developed for configuring and executing simulations on autonomous robot navigation. The simulator can also be used for various types of simulations, including those simulations with autonomous (mobile) intelligent systems in general.

Two different software programs were developed: SINAR, a simulator that shows graphically the representation of the environment and the simulation in real time; and CONAR, the autonomous controller that receives sensor data (from SINAR) and output the actuators data (to SINAR). Simulations with multiple robots can be done if more than one controller (CONAR) connects to SINAR. The communication between both programs is represented on the following figure:

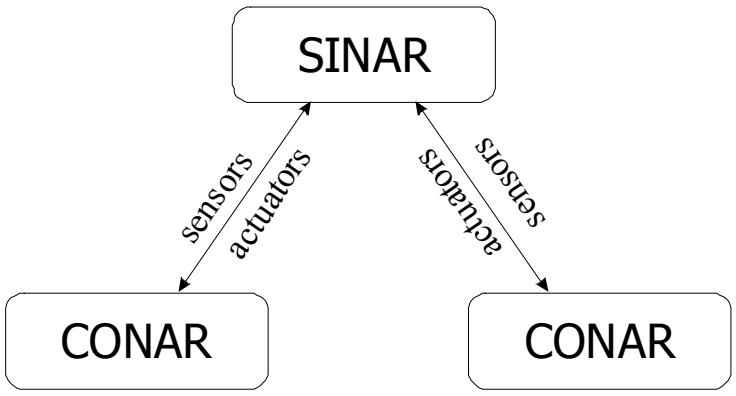


Figure B.1: Communication between SINAR and CONAR.

The communication protocol is implemented using TCP/IP sockets. Thus, several controllers can run under distinct computers over a network (distributing the computing load through different nodes of a network).

Both (simulator) programs were developed under the operating system Linux; the graphical interface was developed with Qt library [36] and some graphical plots were created with Qwt library [37]. The computer language used is C++ language.

Next sections describes each of these programs thoroughly, SINAR and CONAR.

B.2 SINAR (Simulator for Autonomous Robot Navigation)

SINAR is a simulator for autonomous robot navigation experiments. Its graphical user interface contains menus, commands bars, and the environment display (Fig. B.2).

The user can create simulation environments merely by clicking and moving the mouse cursor in the display area. Objects are inserted, resized, translated and rotated simply by using the mouse

device. An object can be an obstacle, a target or even a robot. The user can also edit an object by changing its color and type of movement (for moving objects).

Environments can be saved on files and posteriorly they can be loaded for being used in simulations. Before a simulation starts, one or more controllers (CONAR) should be connected to the SINAR software. The user can control the simulation by activating appropriate (button) commands: start, pause and finish.

During a simulation, sensor data can be viewed graphically through plots in real time (Fig. B.3). The performance of the robot (number of collisions, number of target captures, and number of executed iterations) can be verified in real time as well.

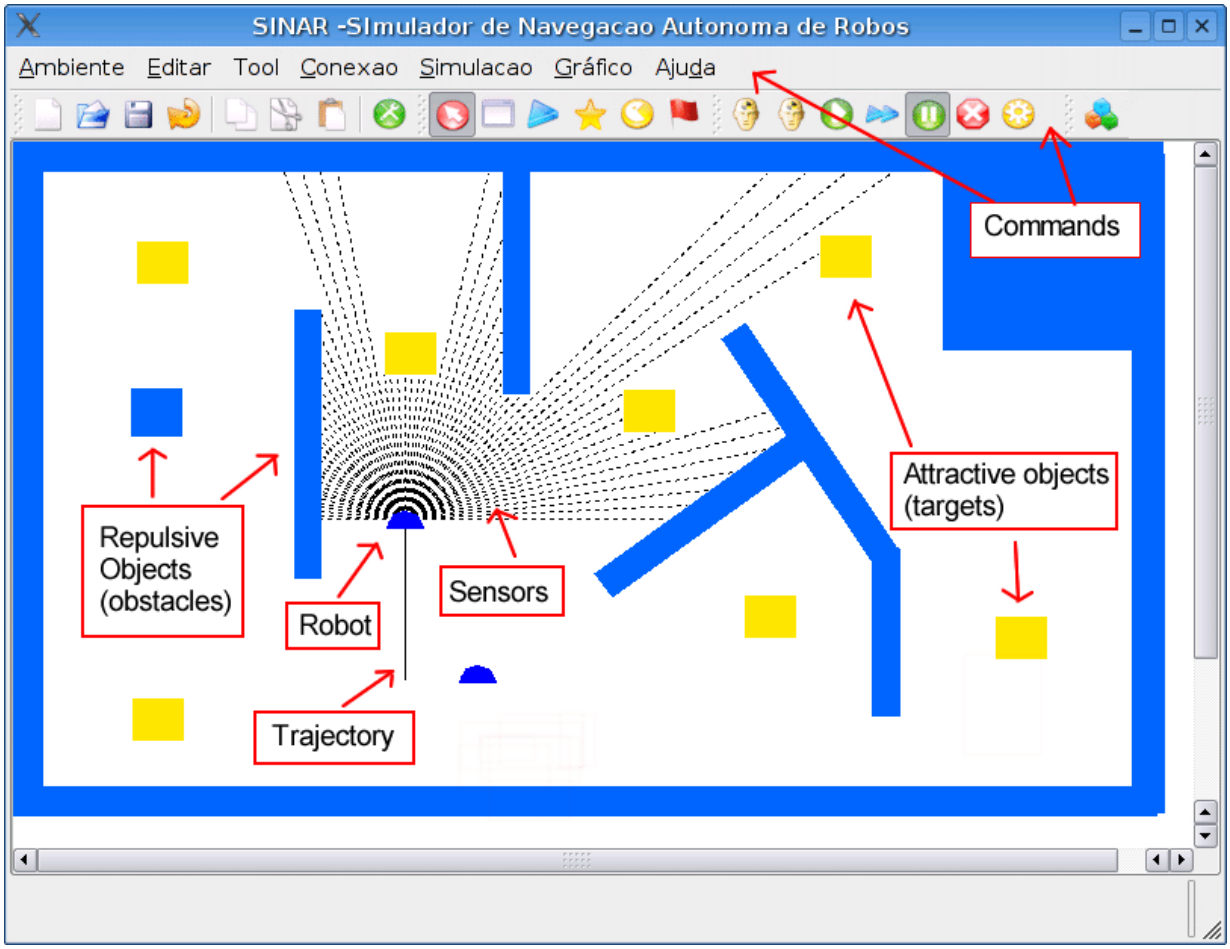


Figure B.2: SINAR interface.

There are two modes of simulation: ordinary mode and sophisticated mode. In the former mode, the environment display is updated at each iteration such that the user can view graphically the progress of the simulation. Furthermore, the user can move any object in the environment in real time.

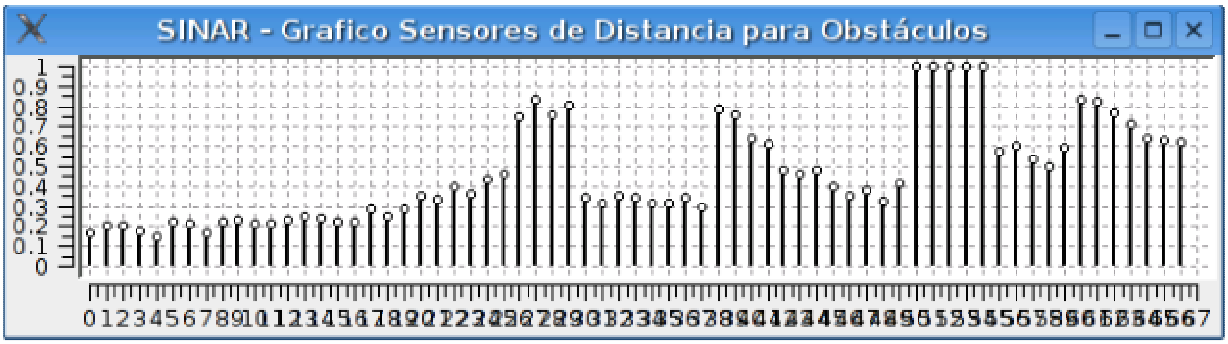


Figure B.3: Sensor data visualization in real time.

In the latter mode, the simulation is accomplished implicitly (not graphically) and is composed of a set of experiments configured by a specific C++ script. The script determines the sequence and the duration of simulation experiments (considering that each experiment uses distinct environments), besides the number of repetitions for a sequence of experiments. In the sophisticated mode, all generated data are recorded in files: the trajectory of the robot and the performance measures (number of captures, collisions and monotonies, and their respective iteration time); the representation of the final state of the environment in PNG format and the performance plot (also called learning evolution graphic) are also generated automatically. The controller data (neural networks states) is also saved in an automatic way since the script tells CONAR to save its state when each simulation is finished.

B.3 CONAR (Controller for Autonomous Robot Navigation)

CONAR is a program that simulates the brain of a robot located in the SINAR environment. After receiving sensor data (distance, color and contact) from its respective robot in the SINAR environment, it sends actuator data (direction adjustment and velocity adjustment) to the same robot. This cycle is kept until the simulations ends.

The graphical interface of CONAR is shown on the following figure. Parameters of the controller can be adjusted before the simulation and in real time; commands can be activated by clicking on buttons: connect to SINAR, apply parameters changes in real time, generate performance data and plots for recording in files, save neural networks state, exit simulation. Furthermore, some neural networks in the controller can be disabled in real time (so that it outputs null (zero)): IP, IC, RR and AR networks.

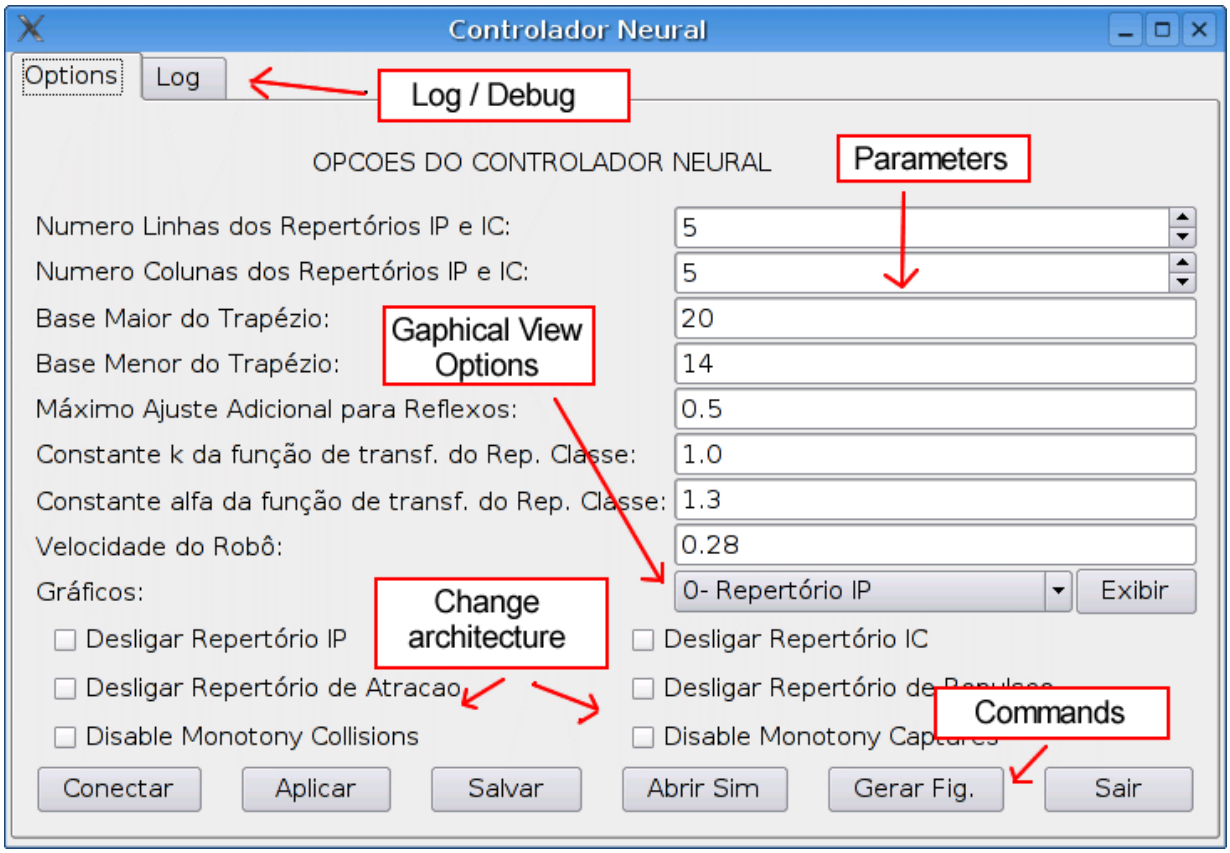


Figure B.4: CONAR interface.

In addition, neural networks state can be viewed graphically in real time (Fig. B.5, Fig. B.6 and Fig. B.7). In the following figures, a neuron is represented by a circle. In addition, the blacker a neuron is, stronger is its output.

In Fig. B.5, it is shown a representation of PI repertoire neurons. A small red square inside a circle means that a neuron has already been winner during a learning event. Figure B.6 shows AR, RR and actuator neurons. The energy levels (degree of activity) of AR or RR neurons are represented by a thick line next to the respective neurons. In Fig. B.7, it is shown the graphical representation of output of winner neurons in PI repertoire (each line represents the winner neuron output in a column: the first line corresponds to the first column and so on).

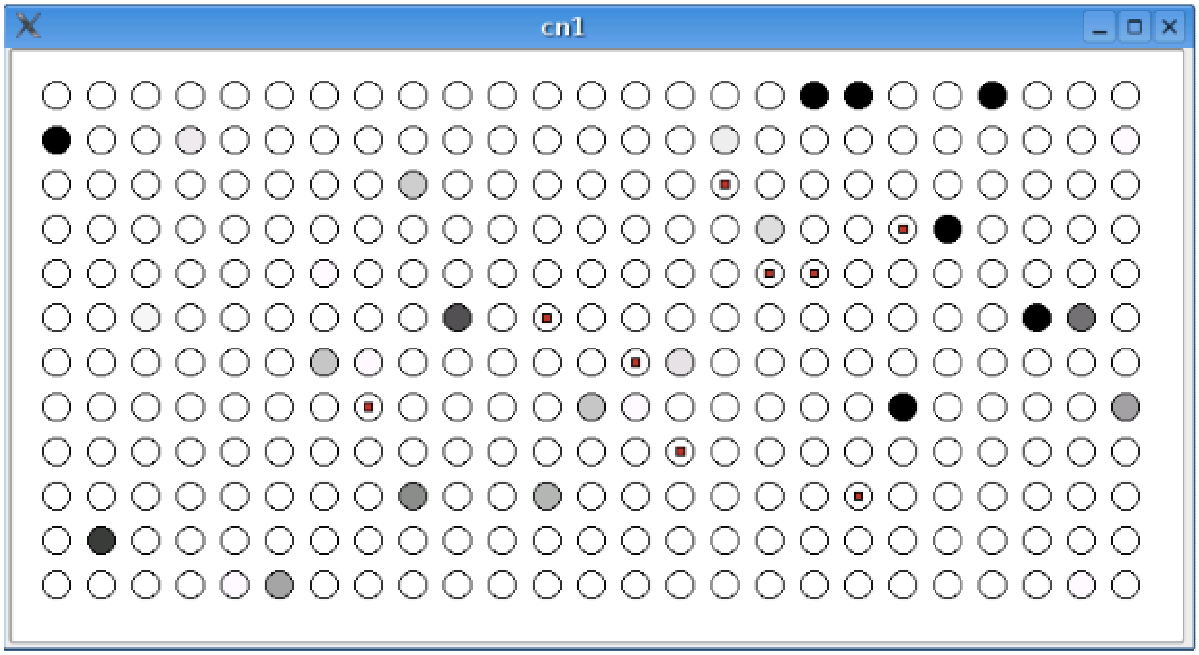


Figure B.5: Graphical representation of PI neural network.

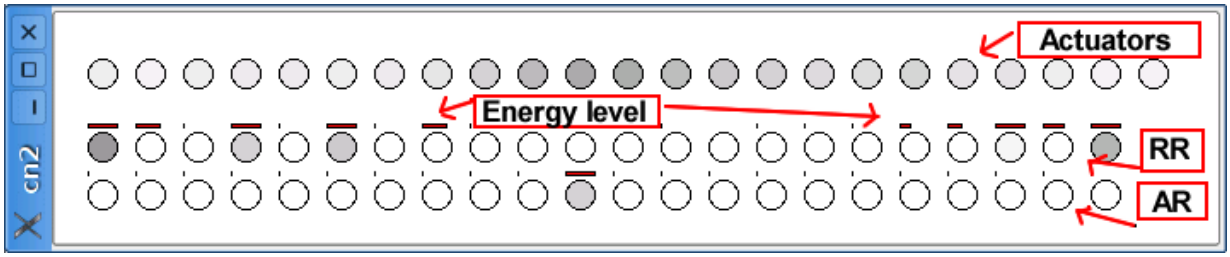


Figure B.6: Graphical representation of RR, AR, and actuator networks.

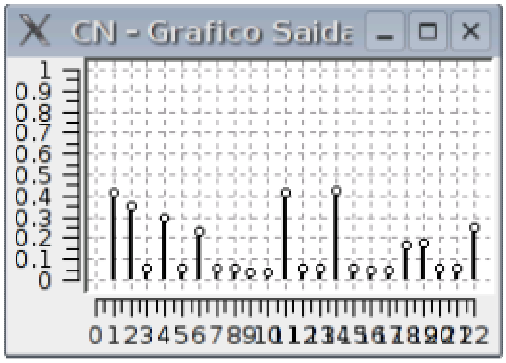


Figure B.7: Graphical view of the output of winner neurons in PI network.

References

[1] Figueiredo, M., *Autonomous Robot Navigation*, Proceedings of VII SBC School for Informatics, Novo Hamburgo, 1999 (in portuguese).

[2] Calvo, R., Figueiredo M., Antonelo, E.A., *Evolutionary fuzzy system for architecture control in a constructive neural network*, Proceedings of the 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'2005), Helsinki, Finland, 2005.

[3] Vershure, P. and Voegtlin, T., *A bottom up approach towards the acquisition and expression of sequential representations applied to behaving real-world device: distributed adaptive control III*, Neural Networks, Vol. 11, pp. 1531 – 1549, 1998.

[4] Crestani, P.R., Figueiredo, M., and Von Zuben, F.,*A hierarchical neuro-fuzzy approach to autonomous navigation*, Proceedings of 2002 International Joint Conference on Neural Networks, (cd-rom), Honolulu, USA, 2002.

[5] Calvo, R. and Figueiredo M., *Reinforcement learning for hierarchical and modular neural network in autonomous robot navigation*, Proceedings of 2003 International Joint Conference on Neural Networks – IJCNN, Oregon, USA, 2003.

[6] Millan, J., *Rapid, safe, and incremental learning of navigation strategies*; IEEE Transactions on Systems, Man, and Cybernetics, Vol. 26, no 3, pp.408-420, 1996.

[7] Cazangi, R. and Figueiredo, M., *Simultaneous emergence of conflicting basic behaviors and their coordination in an evolutionary autonomous navigation system*, Proceedings of 2002 IEEE Congress on Evolutionary Computation, (cd-rom), Honolulu, EUA, 2002.

[8] Kandell, E.; Schwartz, J. and Jessel, T. *Principles of neural science*, Elsevier, New York, 1991.

[9] Donahoe, J. W. & Palmer, D. C., *Learning and complex behavior*, Allyn and Bacon, Needham Heights, Massachusetts, EUA, 1994.

[10] Edelman, G., *Neural darwinism: the theory of neuronal group selection*, Basic books, New York, USA, 1987.

[11] Arkin, R.C., *Behavior-Based Robotics*, The MIT Press, 1998.

[12] Baum, W. M., Understanding Behaviorism: Science, Behavior, and Culture. New York: HarperCollins, 1994.

[13] Brooks, R. A., *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, pp. 14 – 23, 1986.

[14] Colombetti, M., Dorigo, M., Borghi, *Behavior Analysis and Training – A Methodology for Behavior Engineering*, IEEE Transactions on Systems, man, and Cybernetics, Part B, Cybernetics, vol. 26, nº 3, pp. 365-380, junho, 1996.

[15] Haykin, S., *Neural Networks: a comprehensive foundation*, Prentice Hall, New York, USA, 1994.

[16] Kohonen, T., *Self-Organization and associative memory*, Springer Verlag, Heildelberg, 3ª edition, 1989.

[17] Kohonen, T., *Self-Organising Maps*, Springer-Verlag, 1995.

[18] Pavlov, I. P., *Conditioned Reflexes*, Oxford University Press, London, England, 1927.

[19] Rumelhart, D. E., Hinton, G. E. e Willians, R. J., *Learning representations of back-propagation errors*, Nature, Vol. 323, pp 533-536, 1986.

[20] Steels, L. *When are Robots Intelligent Autonomous Agents?*, Robotics and Autonomous Systems, vol. 15, pp. 3-9, 1995.

[21] Thorndike, E. L., *Animal Intelligence*, Hafner, Darien, CT, 1911.

[22] Nilsson, N. J., *Shakey the robot*, SRI AI Center, tech. note 323, Apr. 1984.

[23] G. Giralt, R. Chatila, and M. Vaisset, *An integrated navigation and motion control system for autonomous multisensory mobile robots*, in Robotics Research 1, Brady and Paul, Eds. Cambridge, MA: M.I.T. 1983, 191-214.

[24] Moravec, H., *Towards automatic visual obstacle avoidance*, Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, p.584, August, 1977.

[25] Moravec, H., *The Stanford Cart and the CMU Rover*, Proceedings of the IEEE, Vol. 71, no 7, pp. 872 – 884, 1983.

[26] Mataric, M. J., *Learning in behavior-based multi-robot systems: policies, models, and other agents*, Journal of Cognitive Systems Research, Vol. 2, pp. 81 – 93.

[27] Floreano, D., Mondada, F., Perez-Uribe, A. and Roggen, D., *Evolution of Embodied Intelligence*. in F. Iida, R. Pfeifer, L. Steels and Y. Kuniyoshi (Eds.), Embodied Artificial Intelligence, LNCS serie, Vol. 3139, Berlin: Spriger Verlag, 2004.

[28] Antonelo, E. A., Figueiredo M., Baerveldt, A. and Calvo, C., *Intelligent Autonomous Navigation for Mobile Robots: Spatial Concept Acquisition and Object Discrimination*, Proceedings of the 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA'2005), Helsinki, Finland, 2005.

[29] Skinner, B. F., *Science and human behavior*, New York, 1953.

[30] McCulloc, W. & Pitts, W., *A logical calculus of the ideas imminent in nervous activity*, Bull Math. Biophys., vol. 5, pp. 115 -133.

[31] Hopfield, J.J., *Neural Network and Physical Systems with Emergent Collective Computational Abilities",* Proc. Nat. Acad. Sci., vol. 79, April, pp. 2554-2558, 1982.

[32] Cazangi, R.R., Von Zuben, F.J., Figueiredo, M.F. *Autonomous Navigation System Applied to Collective Robotics with Ant-Inspired Communication*. Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO'2005), Washington D.C., USA, vol. 1, pp. 121-128, 2005.

[33] Mondada, F., Franzi, E. and Ienne, P., *Mobile Robot Miniaturization: A Tool for Investigation in Control Algorithms*. In Yoshikawa, T. and Miyazaki, F (eds.) Proceedings of the Third International Symposium on Experimental Robotics, Tokyo: Springer Verlag, 501-513, 1993.

[34] Nolfi, S. and Floreano, D., *Neural Synthesis of Artificial Organisms through Evolution*. Trends in Cognitive Science, 6, 31-37, 2002.

[35] Steels, L., *The Artificial Life Roots of Artificial Intelligence*, Artificial Life Journal, Vol. 1,1. MIT Press, Cambridge. pp. 75-110, 1994.

[36] Qt library. http://www.trolltech.com/ . Date: December 23, 2005.

[37] Qwt library. http://qwt.sourceforge.net/ . Date: December 23, 2005.