Control system of person following robot: The indoor exploration subtask

Solaiman. Shokur

20th February 2004

Contents

Chapter 1

Introduction

The goal of the main project is to develop a mobile robot able to find, follow, and monitor a human target moving in a domestic environment. This project involves both hardware and software components and is part of the RoboCare Project [1]. The hardware part includes a research for different types of sensors able to locate a distant target, in particular the implementation of a directional radio transmitter and receiver.
On the other side, my scope was to manage the exploration task, that occurs when the robot looses the target. My research includes: definition of the fitness function, research for a convenient neural network architecture, definition of a strategy of evolution and definition of a realistic environment for the simulation.

1.1 An historical overview: Embodied Cognitive Science

The *Embodied Cognitive Science* represents a different and exciting new point of view challenging the traditional cognitivism and connessionism .
The *traditional cognitivism*, born in the middle of the past century, was a response to the *behaviorist* point of view, which described all psychological states as sets of stimuli and the response of the body to those stimuli, defining the mechanism between them as a black box. The cognitivism gives an answer to these black boxes, saying that what is actually between the stimulus and the response is a computer (John Searle: "The idea is that

[1] The goal of the RoboCare project is to build a multi-agent system which generates user services for human assistance. The system is to be implemented on a distributed and heterogeneous platform, consisting of a hardware and software prototype. The project is funded by the Italian Ministry of Education, University and Research (MIUR). It is organized in 3 tasks: 1. Development of a HW/SW framework to support the system 2. Study and implementation of a supervisor agent 3. Realization of robotic agents and technology integration; for more information see http://pst.ip.rm.cnr.it/robocare/

unless you believe in the existence of immortal Cartesian souls, you must believe that the brain is a computer" [4]).
However according to our everyday life experience, great differences appears between a human brain and a computer : if the brain is a computer, why is it, that what is easy for the brain is so difficult for a computer (recognition of an object for example) and, in the opposite way, what is easy for a computer is so difficult for the brain (calculation) (ref. Parisi D., Oral communication).
The classical connessionism gives another answer to the black box: a theoretical neural network expressed as a computer programm. A progress in this direction has been proposed by *Embodied Cognitive Science* considering the agent (with a neural network) and the environment as a single system, i.e. saying that the two aspects are so intimately connected that a description of each of them in isolation does not make much sense [5, 10].
This work uses these new research paradigms, that is Neural Networks, sensory-motor coordination, to satisfy a technological and theoretical problem.

1.2 Reactive, pro-active and Hybrid agents

Typical reactive agents are the Braitenberg vehicles [11], which respond always in the same way to the same sensory input. As we will see this kind of stereotypical behavior is often not satisfying for an exploration task.

On the other hand, recent work have shown interesting results, for tasks similar to our's, with pro-active agents. Pro-active agents actions don't depend only on the sensory input but also on an "internal state" that is defined as the subset of variables that co-determine the future input-output mapping of the agent. Typically, an internal state is realized by a neural controller that translates present and past sensory inputs into actions. Miglino et all have shown recently that introducing an internal state increases significantly performance of an agent that must make a detour around an obstacle to reach a target [17]. Nolfi S. and Guido de Croon demonstrated the importance of internal states for the self localization task [20, 21]. However, these works show also that introduction of internal states raises the level of complexity. Prediction and analysis are very difficult, understanding how internal states are used by the agents is often all but obvious.
Instead, we decided to work on a hybrid form of agent, adding both advantages of the two precedent types: simplicity of analysis of the reactive agent and variability of the behavior of the pro-active agents. We obtain this compromise by implementing reactive agents controlled by: a neural network with a stochastic input, a neural network with an input using an external memory encoding the robot's previous position and a modular neural network.

1.3 Problem Statement

My main work was to solve the problem of finding the target when the robot was lost, i.e when there is no information from the distal sensors (for example radio sensor). In this case the robot should be able to explore the environment until it has the information about the target again.
Many research works for exploration task in robotic use a method called *Internal map*. Basically, the robot evolves in an environment and registers its movements to obtain a map (see section 3.2). This kind of method are not very robust and have many problems in a complex changing and realistic environment. My work was focused on another point of view:

Define a robust strategy, using evolutionary robotics methods, to perform the exploration task with (Hybrid) reactive agents. For this task we will have two constraints:

- *Space: The robot should be able to explore every part of the environment*
- *Time: It has to do it as fast as possible, avoiding as much as possible cycles.*

1.4 Thesis outline

The outline of the thesis is as follows: in chapter 2, we explain methods used for this project as the sampling method and the evolutionary robotics Basic theoretical knowledge as neural networks and genetic algorithms are not presented here. Interested readers could read chapter 2 in "Evolutionary Robotics" written by Nolfi S. and Floreano D. [12] as an introduction to these notions. Following a bio-inspired engineering point of view, we give in chapter 3 an overview of spatial information integration b both in robotics and with animals. In addition another project working on the human target following is briefly presented. In chapter 4, we explain more precisely what we mean by exploration, presenting what makes it complicated. Chapter 5 presents the departure point of my project, where we explain the precedent implementations of the human target following, its results and its limitations, that we tried to overcome in my project. All my experiments and results are explained in detail in chapter 6. Chapter 7 concludes the research, reassuming the most interesting results and the next possible steps to ameliorate them.

Chapter 2

Method

2.1 Evolutionary Robotics using simulator

Evolutionary Robotics, based on the darwinian evolutionary process, permit to find robust robot controllers, difficult to find with handcraft implementation. The process begins with creation of a set of random genotypes defining the weights of the neural networks that control the robots. All these controllers are applied to a task and receive a score proportional to their ability to perform it. At the end of a determinate life-time, we select those who have the best score and use them for the next generation, after having applied the genetic operators (crossing over and mutation). The process continues till having a controller that performs perfectly the task or until a stopping criterion is met.
As an evolutionary process involving real Robots could be excessively expensive in term of time, a simulation approach using a realistic representation of the real robot and of the environment is preferred. The idea is first to let a good controller evolve in simulation and, in a second time, continue the evolution on the real Robot. The simulator used for this project is Evorobot [1] , that permits realistic experiments and the transition to real robots. Following this reasoning, we implement a realistic representation of the robot Koala (that will be used as real robot for the project) on Evorobot.

2.2 Realistic simulation

As experiences on the simulator are a preliminary step before the use of a real robot, there is an obvious interest to make them as realistic as possible. The robot that will be used for the human target following task will be a Koala. The Koala has been preferred to the Khepera robot for practical reasons, as the Koala has bigger dimensions and gives the possibility to use the radio transmitter and receiver. The problem was that in the Evorobot software only the Khepera robot was simulated. To be able to simulate the

Koala Robot, we have done some changes in the code:

- The dimensions of the simulated robot (30 x 30 cm for the Koala, 7 cm diameter for the Khepera) have been changed
- The maximum speed (0.6 $\frac{m}{s}$ for the Koala and 1 $\frac{m}{s}$ for the Khepera) has been modified.
- The sensors have been accurately changed as:
    . The Koala has 16 infrared and ambient sensors, Khepera has 8
    . The maximum range of this sensor is 20cm for the Koala and 5cm for the Khepera

One way to simulate this sensors could be to implement their behavior by using their theoretical specificities (i.e using the non-linear function of light reflected value versus the distance to an obstacle). However, this method appears dramatically irrelevant for simulating realistic cases.
First of all, even if all the sensors of the Koala appear identical, they often response in a different way to the same external stimulus, given their intrinsic properties. In addition, if one compares different robots, one notices that the position and the orientation of the sensors are not exactly the same. Finally, the Koala's sensors are not distance measurements but a measure of the quantity of light that a given obstacle reflects back to the robot. Thus this measure depends of the reflexivity of the obstacle (color, kind of surface, ...), the shape of the object, and the ambient light settings
A more interesting way to simulate these sensors according to their specificities is to empirically sample the different classes of objects that appear in the environment [3].
Using this method, we have sampled three classes of objects : a wall, a small cylinder (diameter smaller than the robot's dimensions: 20cm) and a big cylinder (diameter bigger than the robot's dimensions: 42cm). The real robot was first placed in front of one of these objects (distance 5cm), and then automatically recorded all its sensors for 180 orientations and for 20 different distances. In the simulator, given the orientation and the distance of the simulated robot to a given obstacle, the corresponding line in the sampling file is used as input vector.

2.3 Calculating the relative position: Odometry and magnetic compass

The Koala has an internal counter that calculates exactly how much the motors turn [2]. Using the right and left motor counters, one could determine the position of the Koala at every time step. Unfortunately, even if the internal counter for the motors is exactly known, the calculation for the position

of the Koala cannot be that precise, as the way in which Koala's wheels turn depends a lot on the kind of surface used. For example the Koala turns much more easily on a smooth surface than on a carpet. We have estimated an error rate of about 1% between the position internally calculated by the Koala (based on the motor's counter), and it's real position. Calculus was done as follows: the Koala randomly moves in the environment and registers its position, after approximately 10 meters covered, the robot is asked to go back to the initial position (set as (0.0) position). We calculated then the difference between the effective initial point and this new point. For a more realistic simulation, this error rate was introduced in the simulator when odometry was used.

Our test shows also that the error rate of the calculus made by the robot for its position increases when the robot turns. In fact, tests with a Robot that had to go only forward and come back to the initial point were much more precise (less then 0.5% of error). What increases the error rate is the accumulation of errors on the angle. To avoid this we could add a magnetic compass to the Koala. This kind of component has been successfully used by the K-team on the Koala (ref. Mondada F. Oral communication).

In our case, as we didn't test a magnetic component, we didn't simulate it on Evorobot, we simply assume that having 1% of error on the position is not unrealistic. A more precise and accurate estimation could be done when this this component will be available.

Chapter 3

State of Art

3.1 Human Target following project

Bahadori S. et all, also involved in the Robocare project, are developing robots for the same task of "human target following", but with a very different point of view [23]. Their work is mainly based on 3D reconstructions through Stereo Vision. By computing a difference between a stored image, representing the background, and new images, they isolate the moving objects (as robots and persons) from the environment. The robot is differentiated from persons with a particular marker. Then, by using stereo computation, the position of the robot and the person are found.

The main idea for the next step is to use this information about robots' and persons' position to manage the target following task, with some planification algorithms.
The limitation of such a system are the following:

- The number of persons in the room has to be limited, the moving objects could be of a maximum 3 or 4.
- The person should be easily differentiable from the robot.
- If one wants to cover a big room, or even maybe several rooms, the number of stereo cameras needed could become high.

I think that the third point here is very important, as, although, this approach is highly interesting if one needs a Robot that has to follow the target in a small area, it become really expensive because of the number of stereo cameras involved and the complexity of installation (which is different for every type of surroundings).

3.2 Spatial information in Robots

Robots that have to explore an environment is certainly one of the most classical problems in the filed of robotics . Implementation of robots able to go out of a labyrinth is a typical exercise proposed to students [1] as well as studied by researchers [26]. The most common way to manage the exploration with robots is to use *internal maps*. We will see here the two very general ways of using internal maps: static and dynamic maps.
The static map is certainly the easiest way to manage the exploration problem, but could often be irrelevant. The main idea is to have a geometric representation layer of a particular environment and topological layer. For example, the letters A to F in the geometrical map shown in figure 4.1 are the identifiers of the topological regions which can be seen as nodes of an indirect graph (see appendix A). Using odometry and complex sensory inputs[2] the robot could be able to keep the track of its current position and orientation. Using a list of all unvisited rooms and the paths to join them (given by the topological graph), the robot is able to optimally explore the environment avoiding cycles. This method could be successfully used if one could have a perfect *a priori* knowledge of the environment and also needs a robot able to explore a particular environment (as used, for example, by Johan Bos and all [27]) but cannot be applied if we want to have a general exploration method for *any* kind of environment.

Instead, robots using a dynamic internal map do not need an *a priori* map of the environment, the map is dynamically created by the robot during exploration. Here, the topological layer is created step by step with roughly: recognition of the environment with sensory inputs and odometry. The recognition of the environment implies huge and precise sensory information to be used. The problem that particulary occurs for robots with poor sensory information (as in our case), is that different places of the environment could give the same sensory vector input (known as the aliasing problem [13]), and so it is impossible for the robot to know if it has already been in a particular position or not. The limitation of odometry is that it is very imprecise and sensitive to error cumulation, and the classical way to manage this accumulation is to re-initialize periodically the position at which the robot identifies perfectly a particular position where it has already been, what is limited again by the sensory information poorness. The typical problem that occurs in this kind of method is that one has to stress with two source of information, that may not concord. The robot, calculates it's position with odometry, and sensory inputs are used for the internal map. When robot wrongly think to be in a point where it has already been, and

[1] http://diwww.epfl.ch/ sshokur/projets/matinfo.zip
[2] Often for this kind of method video cameras are used

the sensory input do not coincide with the registered input, it has to re-actualize, or its position (what it think to be it's position) or the internal map. Actually, it's very difficult to decide which one of the two information have to be actualized, as both odometry and environment recognition could be erroneous.

3.3 Spatial information in animals

There are two principal points of view about animals' strategy to manage with the surrounding environment. The main difference between them is about the referential that animals use to integrate information about the environment: **egocentric** or **exocentric** coding.
In a behaviorist point of view, exploration of an animal could be traduced as a simple association between stimulus and responses. For example, the way to go from one initial point to a food zone will be registered by the animal as a series of movement that will be reproduced if we put the animal again in the initial position [6]. The method could be used by the animals even for long and difficult ways by dividing the road in different little subsequences. In this case the information about the environment is coded according to the animal itself : the referential is egocentric. Another use of egocentric referential has been shown by Wehner et al. for the homing navigation of the desert ants *Cataglyphis*, who can explore a long distance and return directly to the nest. They explain that this ants register all their movements during the exploration for food and that they integrate them to derive the direction to go back to the nest.
The advantage of of egocentric reference coding is to limit the information to be registered to only two parameters: angle and distance. However, it is very difficult to know exactly these two parameters as they are extremely sensitive to error cumulation, especially for the angle [8]. To avoid this accumulation different techniques are used by animals: birds for example use magnetic compass for calculating angle during their long migrations. Insects, in particular ants and bees, use sunlight compass derived by the azimuthal position as well as from spectral gradient in the sky. Hartmann and Wehner have shown how path integration based on solar compass is implemented neurophysiologically to perform homing navigation [28]. Rats re-initialize their information when they recognize a particular place that they had already visited [18, 19].
This very easy and rapid strategy is certainly used by animals for some changes of location, but could be irrelevant in a changing environment. For example, if the position of an object is changed or if an object is added, this method could fail.

The second point of view has been proposed by O'Keefe and Nadel in 1978

[14], and is the base of what is called a "cognitive carte". For animals, as well as for humans beings, the part of the brain that manage the space apprehension and representation is the hippocampus, with the *location cellular* as neural support. In the case of rats for example location cellular, are activated when they are in a particular position of the environment. The importance of location cellular has been demonstrated in an experiment involving rats: those who had lesion in the hippocampus were unable to manage with exploration subtasks such as distinguishing visited and non visited ways in a labyrinth [9].

In this case information about the environment is registered referred to fixed reference marks extracted from the surroundings. The animal registers the reciprocal relation between different places. This method is efficient even if some parts of the environment change and in the worst case, as it has been shown for the rats, if the environment changes too much, there is a selective re-exploration. However, it is not yet very clear what are the fixed references to be registered and how these are integrated by animals.

3.4 Exploring an unknown environment

We will discuss here the techniques used in robotics or inspired by nature that could be useful for us. Remember that our task is to accurately explore any environment, avoiding as much as possible cycles.
We have seen classical solutions used in robotics: use of a static or a dynamic internal map. The first technique has to be immediately rejected, as it is not a general way to explore any environment.
We have to be careful not to confuse dynamical internal maps in robotics and cognitive maps in animals, as internal maps could be performed with egocentric or exocentric information (contrary of animals where the notion of cognitive map is associated only with exocentric coding[3]). An exocentric registration of the surroundings implies the recognition of particular places in the environment and the relations between them. This method requires a huge amount of information about the environment [31]. For example for rats, which recognize some parts of the environment (proved by the fact that there are specific location cells that are activated only when the Rat is in a particular place [14]) there are a lot of different sensorial modalities used : visual seems to be the most important, but audition and smell are also involved (as shown by Hill and Best, even blind rats use location cellular [22]). So there is a huge amount of information (think about vision that involves color, distance, shapes, ...) used to detect unambiguously a particular position of the space. Compared to the rat, our sensory information (16 Infra-red proximity and ambient light sensors) is dramatically

[3]if we accept the definition given by O'Keefe and Nadel [14]

poor. Robotic projects using these kind of techniques are confronted to a dilemma: augmenting the sensory information by adding video cameras and techniques of image analyze that slow down the robots' behavior (see 3.1), or limit the exploration task to very simple environments (see [26]). We would like to avoid both disadvantages.

In the other hand we have seen that the problem of egocentric information is the possible cumulation of errors, that could be corrected with external reference: for example sunlight for insects, environment recognition for Rats or magnetic compass for birds.
A solution inspired by insects has been successfully implemented by Kim D. and Hallam J. C. T. [30] on a simulated Khepera. They have shown how a robot can use a referential light source, as a lamp, to perform homing navigation. However, as in our case the robot has to explore more than one single room, this technique cannot work.
The problem of landmark recognition is the same as in the exocentric coding: it implies more sensory input information to avoid aliasing problems.
Instead, a method inspired by birds using a magnetic compass could be more interesting, we could use it compounded with odometry to avoid angle error cumulation and having an egocentric information about the environment.
Thus we decided not to use internal - dynamic or static - maps. We tried to find other kinds of techniques using egocentric information, in a more easy and less dependant on error way (see in section 6.4) or even technics not using at all egocentric information (see section 6.3 and 6.5).

Chapter 4

Exploration

4.1 Why is Exploration ability useful for this project

The ability to correctly explore an environment is very useful for a robot which has to follow a human target. We propose here two situations where this ability occurs:

- When the robot looses the target, i.e when it has no information from the distal sensors (example radio sensor) because of the distance.
- When there is an obstacle between the robot and the target that has to be circumvented (losing may be temporarily the information from the distal sensor)

In these two cases, the robot has to explore the environment till it has again information about the target.

4.2 Formalization of the Exploration task

In our study, we will split the problems in two sub-problems: following the target and exploring. By exploration, we mean that the robot should be able to visit any part, or any rooms, of any realistic environment. We can formalize the problem as a problem of research on an undirect Graph $G(V, E, \Psi)$ (see Appendix A), where V (node) is the set of all rooms of the environment, E the set of all doors (see figure 4.1), and Ψ the function that associates for each door the two rooms separated by the corresponding door.

Thus our problem could be formalized as Graph Searching with visit of all nodes and the avoidance of cycles as constraints.

Notice that the set of environments E_V that we will consider, are those who's corresponding graph $G(V, E, \Psi)$ is connected. That means that the

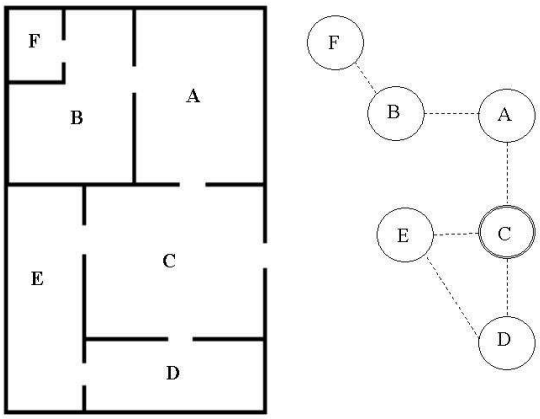


Figure 4.1: (Right) Environment E_0 defined with rooms A,B,C,D,E,F; (Left) Graph $G(V, E, \Psi)$ of E_0: where $V = \{A, B, C, D, E, F\}$ the set of rooms in E_0 and $E =$ the set of doors

excluded environments, are those, very degenerated, that have rooms that cannot be joined by the other rooms (example rooms without door).
We will define a strategy as **acceptable** if one who follows it is able to visit *at least* one time every room and we will say that a strategy is **optimal** if it permits to visit all rooms avoiding any cycle.

So we have basically two different sorts of constraints: space (the exploration strategy should permit to go in every room) and time (it should do it as fast as possible). In our case, it is highly unrealistic to look for an **optimal** solution. Instead, we will say that a strategy is **satisfying**, even if it is suboptimal, if it is acceptable and avoids *as much as possible* cycles.

4.3 What is a complicated Environment?

One of the most challenging problem was to define a good environment, representative of realistic complications that occur in real environments, what is quite different from what appears intuitively complicated.

4.3.1 Internal Rooms

Let us consider a labyrinth[1] that could appear as being *complicated* (figure 4.2).

[1]from http://www.BillsGames.com/mazegenerator

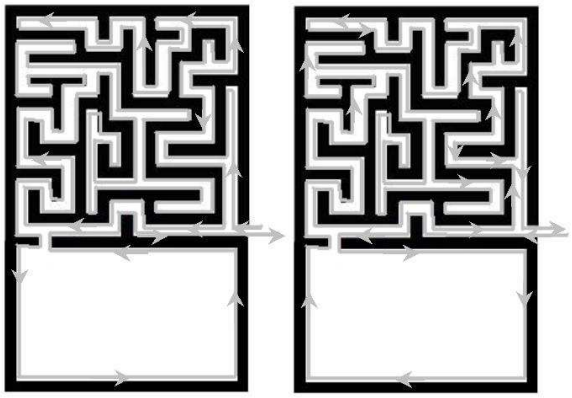


Figure 4.2: a very *common* labyrinth, (Left) a strategy of Left wall following; (Right) a strategy of Right wall following

If one wants to visit the bottom room (and so wants to go out of this labyrinth) without any *a priori* information about the labyrinth's configuration he can choose between two very easy strategies : Left wall following (figure 4.2 (Left)) or Right wall following (figure 4.2 (Right)). The two methods reveal to be successful; obviously one could argue that in this case, following the left wall is a much more optimal solution than the other one, but an agent with no *a priori* knowledge of the environment cannot know it.

However, we don't consider this as a *complicated* environment. Let us consider another labyrinth inspired by the CNR[2] building (figure 4.3). In this case, if the robot is lost (no radio information) and does a Right (or Left) wall following, while the target is in the room A, the robot will never find the target.
So a complicated environment for an exploring task could be defined as an environment with some *internal rooms*. The idea is that the first environment could be integrally explored with simple reactive agents as we will show, but the second one needs a more complex behavior including some changing strategy and\or some information about the environment itself (that could be collected during the life or given as an *a priori* information by the engineer).

We can notice that environments with *internal rooms* are very realistic (not only because it appears in the CNR building). In fact, considering even a normal room with an object stored in the center (for example a bed), the exploration with a wall following will not always have success.

[2]Consiglio Nazionale delle Ricerche, Rome

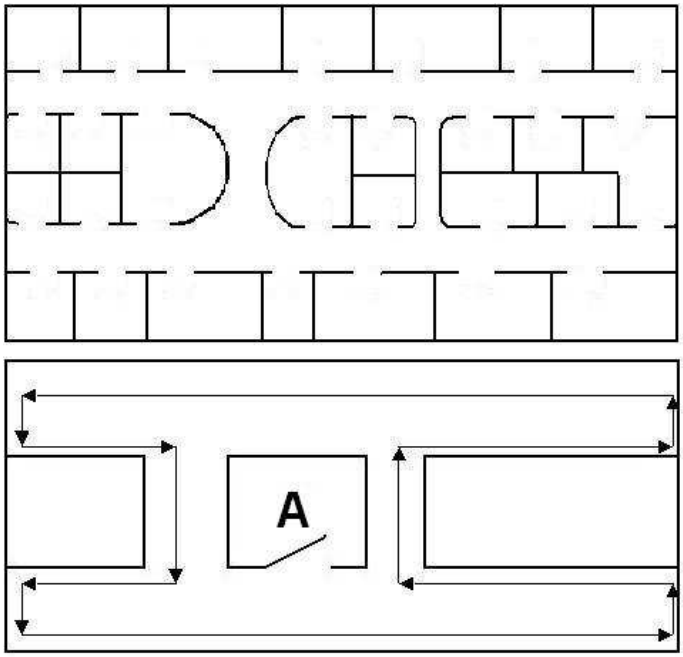


Figure 4.3: [Up] schematic plan of the CNR; [Down] Considering that the target is in the room A, the robot will never find it by doing a simple Right wall following

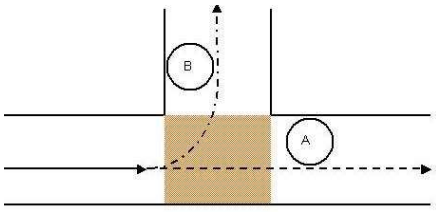


Figure 4.4: Aliasing problem during exploration: when the robot is in the shaded part of the corridor it has to go one time straight on(A)and the other time to the left (B).

4.3.2 Two types of aliasing problems

A second source of complication for exploring an environment is the aliasing problem [13]. An aliasing problem occurs in our case if we have one single place with at least two different possible ways or if there are two (or more) different places that give the same input vector but need different responses (example one time going right, another time going left).
An example for the first aliasing problem could be a normal corridor. As we could see on figure 4.4, when the robot is in the middle of the corridor (shaded), it has two different possible choices. To be able to explore all the environment, it should certainly go at least one time in each direction (A and B). We could notice that this aliasing problem occurs if and only if we have an *internal room* as shown in figure 4.5.
The second aliasing problem is also particulary present in our case because of the dramatically poor sensory information. As said we have omly 16 infrared ambient sensors.
These two types of aliasing problems complicate the exploration task as follows:

- The first type of aliasing renders the exploration task complicated in terms of space. We will show that a simple reactive agent is not able to visit an environment integrally if the first aliasing problem occurs.

- The second type of aliasing renders the exploration task complicated in terms of time. In an environment without any second type aliasing, an agent can be evolved to recognize exactly if it has already been in a particular point of the environment and could easily avoid cycles (by for example changing its direction or strategy of exploration).

In other words, an easy environment can be entirely explored by a simple reactive agent (which always responds in the same way to the same sensory

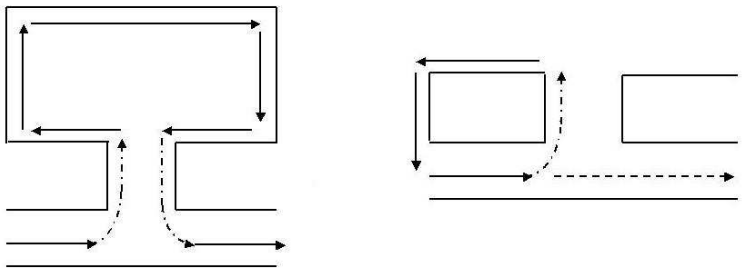


Figure 4.5: (Left) if there is no internal room, by using a simple wall following a robot could explore both direction in a corridor (Right) The robot cannot visit both direction with a wall following, the robot should have two different behaviors for the same sensor information

inputs). As defined above, an internal room gives another level of complexity that prevents simple reactive agents to solve the exploration task.
However, we will see that there are different ways to define an environment with an internal room that gives also a different level of complexity (see section 6.3).
As we will see, another important parameter for the complexity of an environment is the size. The main problem with a big size environment is to manage the cycles, because there is a direct correlation between the size of cycles and the size of memory required. It is intuitively obvious that if one wants to perceive a cycle, one should be able to detect some kind of periodicity, and the more the period is long, the more memory is used to detect it.

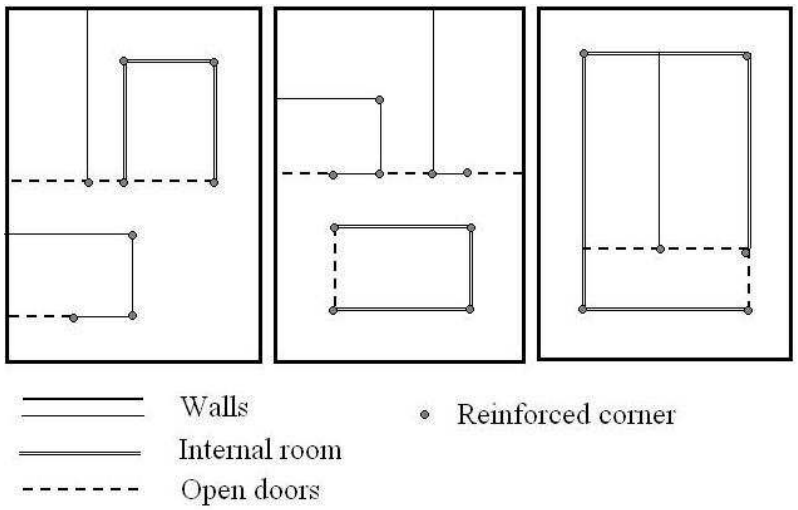


Figure 4.6: Three environments used for our experiments, the little round in the corners are artificially added to give thickness to the walls;(Left)Environment 1; (Middle)Environment 2; (Right)Environment 3

Chapter 5

Precedent work on this project

Before my project, some tests were done for this specific task. We will see in this chapter the results and some limitations that appear in those experiments.

5.1 The transmitter-receiver Device

To allow a robot to localize a distant target, even separated by an obstacle, a radio type transmitter-receiver system consisting of a transmitter carried by the target person and a receiver installed on the robot is under development at the CNR by Raffaele Bianco, Massimiliano Caretti and Stefano Nolfi [16]. This new sensor will permit the robot to find the *relative direction* of the human target. Preliminary tests of this system indicate that it provides reliable information about direction but not about the distance of the target. The system consists of a transmitter producing a continuous signal of 433 MHz (figure 5.1 Left), a receiver (figure 5.1 Top-Right), and a directional antenna (figure 5.1 Bottom-Right). To provide information about the current direction of the transmitter the antenna should be mounted on a motorized support (still to be developed) that allows the rotation of the antenna and the detection of its current orientation with respect to the frontal direction of the robot.
A second more precise sensor that could give additional information about the distance of the target is also under development at the CNR. The main idea is the same as for the first one but it uses sound instead of radio waves. The system consists of a transmitter (on the target) producing very short sounds separated in time and a receiver (on the robot) provided with two omni-directional microphones that detects the arrival of the first waves and then stop listening until echoes have disappeared. The receiver device detect the time difference between the signals detected by the two microphones that

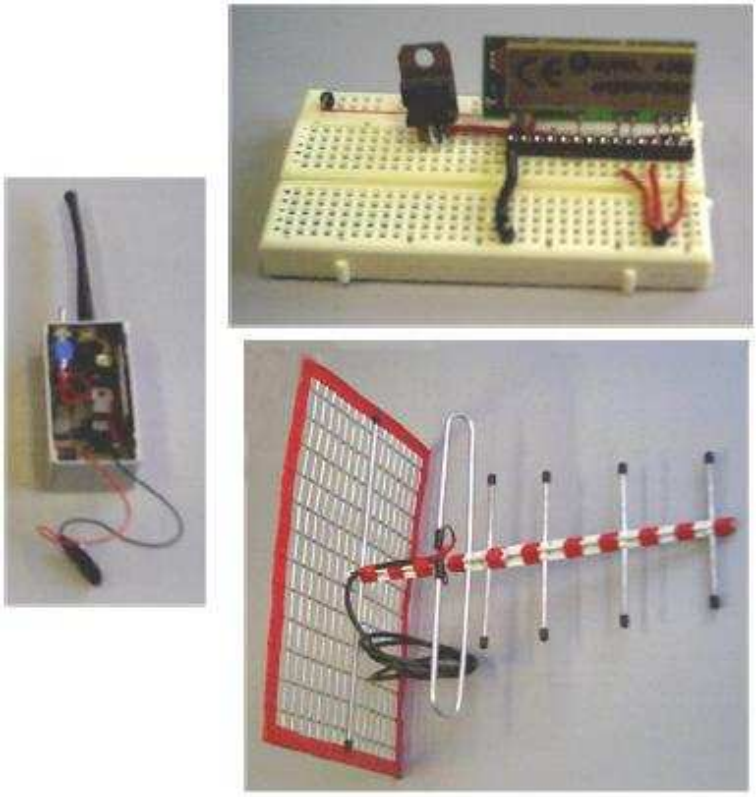


Figure 5.1: The radio transmitter-receiver device. Left: The transmitter. Top-Right: The receiver. Bottom-Right: The antenna.

provides information about both direction and distance.

5.2 Implementation of the human target following robot

The human target following robot was already implemented in the simulator Evorobot by Raffaele Bianco at the CNR (see figure 5.2). Here are the new components added by him into the Evorobot simulator:

- A moving target has been implemented. This agent moves freely in the environment. It avoids obviously the obstacles and at each step he randomly it choses between turning right or left, going straight, accelerating, decelerating or even not moving at all.

- A directional sensor (see section 5.1) has been implemented. This component is a very simplified implementation of the radio transmitter and receiver. As the knowledge about the real radio transmitter and

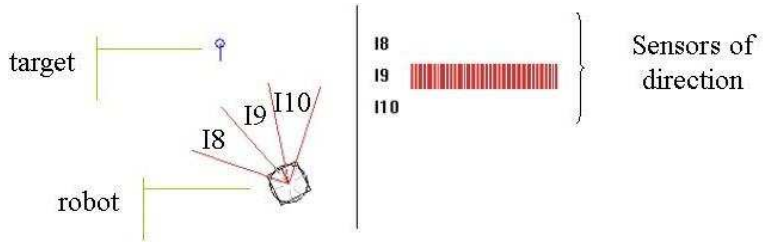


Figure 5.2: Koala Robot and the directional radio transmitter/receiver simulated on Evorobot, the receiver divides the environment in three zone corresponding to three input of the robot, if the target is in one of these zones (no matter its' distance) the corresponding input is activated

receiver component is low, the simulator includes simply three inputs that divide the space in front of the robot, and which are activated if the target is in the corresponding section (see figure 5.2).

One could argue that this simulation is not realistic at all as it doesn't manage any distance problems, in particular there is no attenuation of information if there is an obstacle between the robot and the target or if the target is very far. In addition, in the real case, we could have some interference problems with the radio waves in a room, or attenuation of the signal if there is an open window or a closed door,... All this data will be known only when the radio sensor will be completely implemented, tested and sampled to be used by the simulator. Therefore we decided to concentrate our work on more theoretical problems, that are useful for the next step of the project.
In fact, even if we don't know exactly how the sensor reacts, there is at least one certainty: in some cases the robot will be lost. For example if the target is too far or if there is an obstacle between the robot and the target. We don't have to know if *too far* means exactly 20.5 or 30.4 meters. In this case the robot should be able to explore the environment to find the target (or for having some radio information again).

5.2.1 Experiments Setup

We will see in this section how the first experiments done by Raffaele Bianco were implemented.
The fitness function defined for this first experiment was:

$$1/\text{distance between the robot and the target} \tag{5.1}$$

This function has some limitations:

- As it is not possible to know what is the maximum value: interpretation in term of optimal performance is not possible.

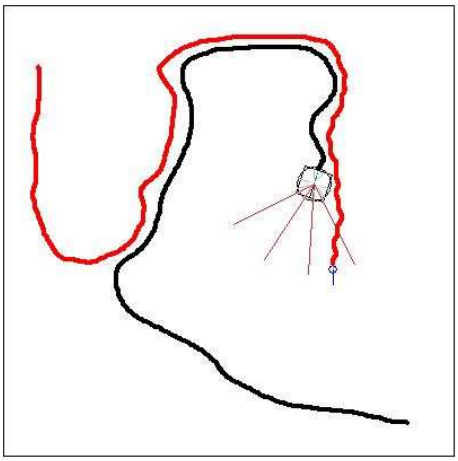


Figure 5.3: Evolved agent following the target, Simulated Koala Robot with the directional radio sensor, The evolved agent tries to minimize his distance with the target

- It is an external function : the variable distance between the robot and the target is not available to the robot it self, and could require complex machineries and procedures if we project to evolve a real robot (p.64-66 in [12]).
- The evolved individuals are not able to follow the target in *all cases* as we will see in the next section.

The architecture of neural network was a standard feedforward , with 11 input i.e. 8 infra-red proximity sensors of the Khepera (the Koala's sensors weren't yet sampled) plus 3 inputs of the radio type sensors and 2 output for Khepera's motor left and right controllers. The environment was the one shown in figure 5.4.

5.2.2 Results and Critics

In the very first part of the project I have tried to identify situations that evolved agents, found with this first implementation, did'n't manage.

The best individuals of these first experiments show a very good ability to follow the target in an easy environment, defined as a box with no obstacle (see figure 5.3).
Using a more complicated environment with obstacles, the agent shows still good abilities to follow the target. In fact in some of these evolved agents there is emergence of wall following strategy when there is an obstacle between them and the target.

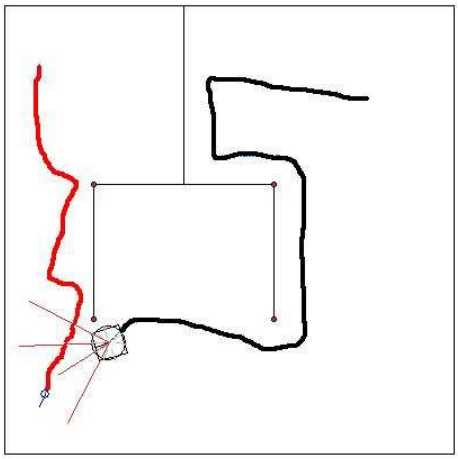


Figure 5.4: Evolved agent following the target, with wall following strategy when there is an obstacle

As shown in figure 5.4 this simple implementation gives a very good result. The main idea is that till the directional radio receiver has information (the directional receiver oriented in the direction of the transmitter), the robot tries to minimize the distance with the transmitter (target), but when this information is lost, it changes its strategy and begins a wall following. However, the fact that sometimes the robot looses the radio information depends on the target movement. For example: the robot is against the wall, it tries to go in the target's direction but an obstacle prevents it. Then, the target moves away and robot's receiver loses the radio information, the robot stops trying to go in a particular direction and follows the obstacle until it has again some radio information available. We could see on figure 5.5 what happens if the target does not move and the radio information remains always available to the robot. In those cases the agent does not go into a wall following strategy and continues indefinitely with the first strategy (minimizing the distance).

However the problem of the blocked robot when the target is not moving is actually not a very serious one as we could resolve it by adding some random movement to the blocked robot (it this case the robot will lose the direction of radio signal and begins a normal wall following).
A much more serious problem is to define a good exploration strategy that could be used by the robot when it looses completely the radio signal and the human target. We have seen that the only strategy used by the agent to explore the ambient, when looking for the target, is a wall following, that is not enough for an *acceptable* exploration of any realistic environment (see section 4) .

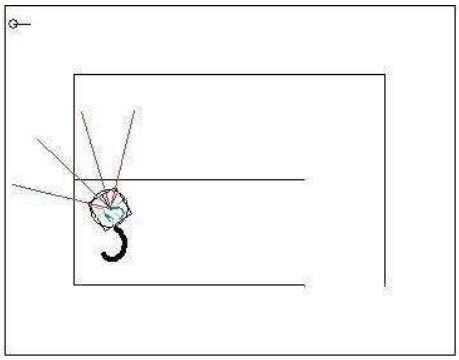


Figure 5.5: If the target (up Left) does not move and if there is an obstacle between the robot and the target, the robot is blocked indefinitely against the wall

Chapter 6

Experiments

In this section I will explain the experiences I have done to find agents able to explore any realistic environment. The strategy was to evolve this agents and to test them on some very *general* and some *complicated* environments. The definition of what is *general* and *complicated* was a difficult task. In fact I had to define environments for the testing part, that ensure us that if the evolved agent was able to explore the totality of them, this agent will be able theoretically to explore *any* environment. The environments used are those defined in section 4 (Environment 1,2 and 3).

6.1 Experiments Setup

As the implementation of the simulation for the Koala was done during the last part of my project (the real Koala Robot was available only the last month), a huge part of the experiments has been done with the simulated Khepera. Then when the Koala was correctly implemented some of the most interesting experiments were reproduced. Obviously, when we compare different types of agents between them and give statistics of success rate, we use only one type of simulated Robot.
Evolution and tests for the next sections was done as follows: Agents were evolved on Environment 1 (see figure 4.6), each experiment was reproduced at least 10 times. A population was composed of 100 randomly generated individuals, and was evolved for at least 300 generation. Individuals' life lasted 5 epochs of 5000 life steps. At the beginning of each epoch, the individual was positioned randomly in the environment.The mutation rate was 4%.
Then the best individual of the best seed was tested 20 times on Environments 1, 2 and 3. At the beginning of each epoch the start position was randomly selected. The life for the test was 20'000 time step, considering that an agent with an optimal strategy should be able to visit the environment integrally in approximatively 4'000 time steps.

Fitness Function

The fitness function was defined as follows:

$$Fitness = \phi + \psi \tag{6.1}$$

Where ϕ is the obstacle avoidance function as defined by Floreano D. and Mondada F. [15]:

$$\phi = (V)(1 - \sqrt{\Delta V})(1 - i); 0 \leq V \leq 1; 0 \leq \Delta V \leq 1; 0 \leq i \leq 1; \tag{6.2}$$

With V: the sum of rotation speeds of the two wheels, v: the absolute value of the algebraic difference between the signed speed values of the wheels (positive is one direction, negative the other), and i: the normalized activation value of the infrared sensor with the highest activity.

And ψ defined by:

$$\psi = \begin{cases} positivescore, & \text{if the agent cross a door for the first time;} \\ 0, & \text{every } par \text{ time that the agent crosses the same door;} \\ negativescore, & \text{every } odd \text{ time that the agent crosses the same door} \\ & \text{and if } \mid (fitness - negativescore) \mid \geq 0. \end{cases} \tag{6.3}$$

The point was to force an individual to go into all rooms (+ score to cross a door) but to avoid to return into the same rooms. As par times are those where the agents simply get out of a visited room, no positif or negative score was given[1]. Obviously, if all rooms were visited the list of the visited rooms was re-initialized (the agent could go back into a room that was already visited and have a positive score).

Many values of the positive and negative score have been tested. The most interesting solutions were obtained with a half incremental score:

- positive score = $5'000\times$ number of visited rooms (+5000 for the first crossed door, + 10'000 for the second one, ...)
- negative score = $-5'000$

[1]Example: a agent crosses a door and goes into a room ⇒ positive score, it recrosses the same door to get out of the room ⇒ no positive or negative score, it recrosses again the same door ⇒ negative score, etc

The reason of an incremental positive score and a normal negative one, was that we prefered an agent a which visits three rooms and returns back in two of them: $fitness(a) = \sum_{i=1}^{3} i * 5'000 - (2 * 5'000) = 20'000$, to an agent b which visits only two rooms $fitness(b) = \sum_{i=1}^{2} i * 5'000 = 15'000$, because agent a explores more space than agent b. Instead a normal positive and negative score would give best fitness for agent b than for agent a: $fitness(a) = 3*5'000-(2*5'000) < 2*5'000 = fitness(b)$ and an incremental positive and negative would give the same fitness for both: $\sum_{i=1}^{3} i * 5'000 - \sum_{i=1}^{2} i * 5'000 = \sum_{i=1}^{2} i * 5'000 = 15'000$.

6.2 Simple reactive agents

For these experiments, I have used some parts of the implementation done by Raffaele Bianco (see 5.2) without using the simulated radio sensor.
The first architecture tested was a simple feedforward with 16 input neurons (Koala's sensors), 4 hidden neurons and 2 output neuron (Koala's motor left and right controllers).
As previewed these simple reactive agents were only able at most to perform a wall following and they never explored integrally an environment with internal rooms (figure 6.1 (Right)). In addition, another problem that occurred here was that sometimes the agent could be blocked in a smaller cycle (figure 6.1 (Left).
In the next tabular the percent of complete exploration, of crash and partial exploration is reported. A partial exploration appears when the agent is either or in micro cycles (in this case it visits often only one room) or in macro cycles. The results are based on 20 tests for each Environments.

	$Environment1$	$Environment2$	$Environment3$
completely visited	0	0	0
crash	0	0	0
partially visited	100	100	100

Change the obstacle avoidance fitness

We have defined our fitness function using the obstacle avoidance fitness Φ (see equation 6.1), which gives a good score for individuals that have tendency to go straight (parameter ΔV). In our case it could be desirable to have an agent that turns more, and that instead of going straight into into corridors has the tendency to turn and enter in the different rooms. So we have changed the fitness function to give less importance to ΔV compare to the parameter speed V and the distance to the walls i:

$$\phi' = (V)(1 - \varphi(\Delta V, k))(1 - i) \tag{6.4}$$

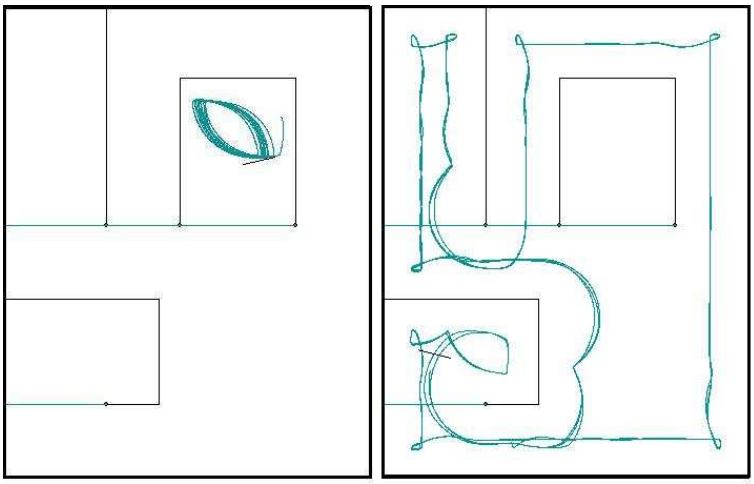


Figure 6.1: Typical behavior of a reactive agent: (Left) Micro-cycle (Right)Wall following and macro-cycles

With V, ΔV and i defined as in the normal obstacle avoidance fitness ϕ and $\varphi(x,k) = \frac{x+(k-1)}{k}$.
Test with three different values of k: 5,10 and 15 have been done. Once again the architecture was a simple feedforward neural network with 16 input neurons (Koala's sensors), 4 hidden neurons and 2 output neuron. Here are the results in function of the value of k (k=0 is the same agent as the one presented in chapter 6.2).
The percent of tests in which the agent has respectively explored all the environment, crashed or explored only one part of the environment is reported in the next table. The results are based on 20 tests for each value of k and each Environments, the value is the average result for Environment 1, 2 and 3.

	$k=0$	$k=5$	$k=10$	$k=15$
completely visited	0	6	4	0
crash	0	0	0	5
partially visited	100	94	96	95

We could understand these results by comparing the path of the best individuals of the best seed for each experiment. The main difference is about the angle between the initial direction and final direction when the agent avoids a wall. The more k is high the more this angle is near of 180°, as we could see on figure 6.2. This ability to change direction could be more or less benefic. In fact, as we have seen results shows that for $k = 5$ the rate of complete exploration raise. The main reason is that, instead of doing a wall following (see $k = 0$) the new agent changes it's direction and can sometimes enter in the internal room, as suggested in the figure. In the other hand what

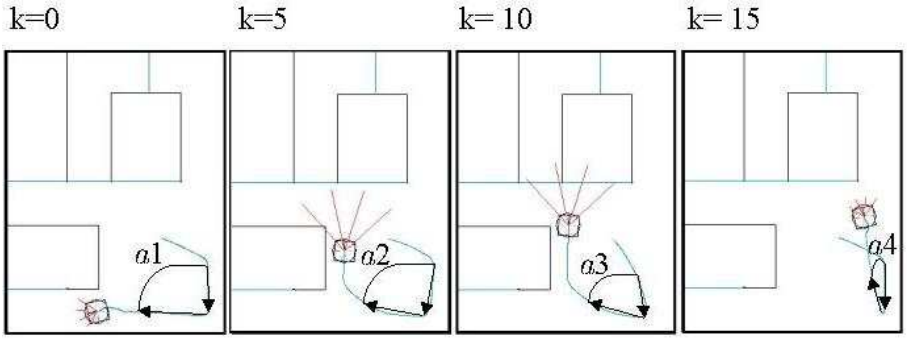


Figure 6.2: Tests of agents evolved with different values of k for the fitness function using ϕ'. $k = 0$ is the normal obstacle avoidance fitness. There is an inverse correlation between the value of k and the angle of the agent when it avoids an obstacle: $a_1 > a_2 > a_3 > a_4$

makes agents with high values of k inefficient, is the fact that, when they avoid an obstacle, they turn completely (often 180°) and by doing so enter in a cycle (they actually go up and down in the right corridor). A good value of k seems to be between 5 and 10, but the optimum value depends a lot of the considerate environment, and we are not interested by finding it. The only important result is that, here we have raising of the success rate for the exploration without a raising of crash rate. Even if in the best case ($k = 5$) the rate of entire exploration is very low, the introduction of the constant k stays interesting if it's coupled with other methods. For next experiments when we have to use the obstacle avoidance fitness function, we chose ϕ' (see equation 6.4) with $k \simeq 5$ instead of the normal ϕ (see equation 6.2).

6.3 Stochastic Neuron

We have seen that the problem with reactive agents was that they give always the same response(motor action) to the same stimulus (sensor input), and we have also seen that in general, there are positions in the environment that require more than one action (see first aliasing problem in section 4) to perform an acceptable exploration.
In this section we will introduce a way to solve this problem by introducing more variability in the agent's behavior, i.e. having agents that don't follow always the same path. The problem that will appear is: if the behavior is too variable the agent will often crash and a lower level of variability will not be enough to explore all kinds of environments. In addition, even if we find an agent who explores an environment integrally , the strategy will be sub-optimal in term of time.

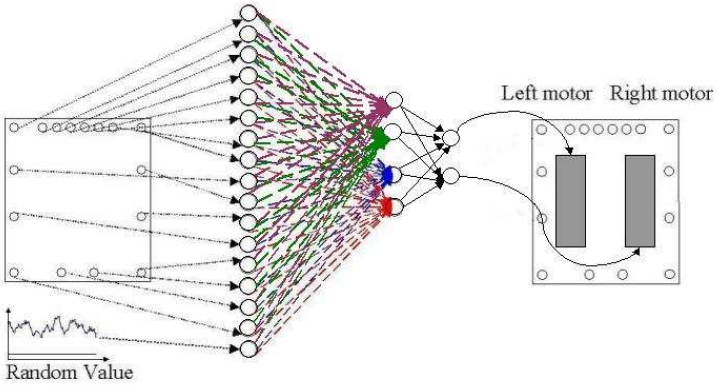


Figure 6.3: A fully connected neural network with a stochastic neuron; Input: 16 normalized infrared sensor of the Koala plus one stochastic neuron; 4 hidden neurons; Output: 2 neurons that control the left and right motor controllers

"A man of genius does't make mistakes,
his mistakes are deliberate and are
gates of discovery"
Philippe Sollers[2]

The idea, to develop an agent who has a more variable behavior was to introduce a stochastic neuron. The architecture was as shown in figure 6.3: we had the 16 infrared sensors of the Koala as input, and in addition we had one stochastic neuron with random values. We have implemented the stochastic neuron as follows:

$$\text{Stochastic neuron[t]} = k*\text{random value[t]}+(1-k)*\text{random value[t-1]}; 0 \leq k \leq 1 \tag{6.5}$$

The parameter k, gives somehow the level of randomness of the stochastic neuron, a value near to 1 gives a very variable neuron (and as we will see also a very variable behavior); contrary k near to 0 gives a neuron with values around 0.5 and few variability and $k = 0$ gives a normal reactive agent as those seen in section 6.2.
The statistics of the experiments with different values of k ($k = 1.0, 0.75, 0.5, 0.25, 0.0$) and different sort of environment (Environment 1,2 and 3) are reported on figure 6.4.

[2]Gisèle Freud, Philippe Sollers, *Trois jours avec James Joyce*, 1982 Denoël, Paris

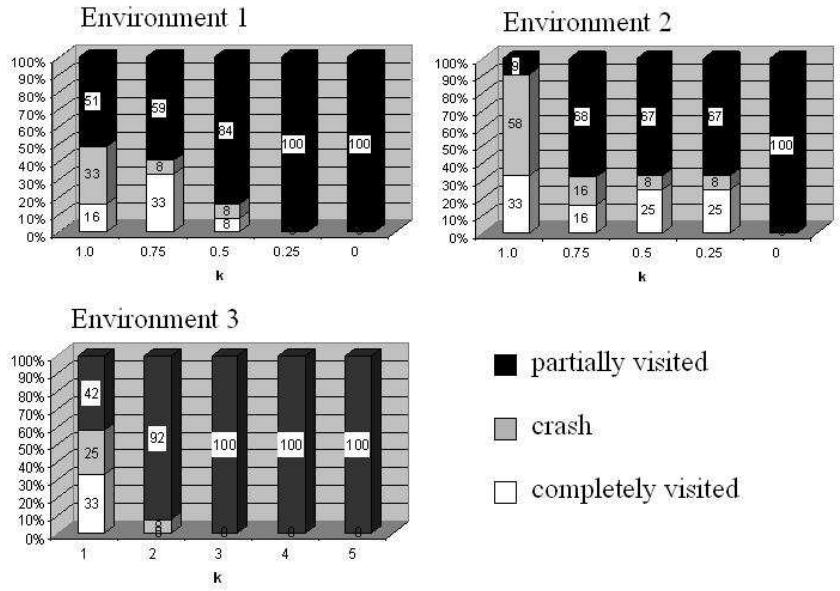


Figure 6.4: Agents with a stochastic neuron, tested on Environments 1,2 and 3; X axis: value of the constant k ($k = 0$ corresponds to the reactive agent) Y axis: type of exploration, Z axis: percentage of experiment (in basis to 20 test for each type of experiment)

An accurate analysis of the statistics reveals that:

- The best agents are able, at most, to visit an environment integrally, in a third of case.
- The rate of acceptable tests and the rate of crash is correlated (correlation for the three environment = 0.59). That means that if we look for agents able to visit all the rooms by adding some kind of variability on its behavior, we augment its probability to crash
- A "good" value of k depends of the environment. For example $k = 0.75$ gives the best results for the first environment (33% of case it visits the environment integrally) but reveals to be inefficient in the third environment (100% of case it visits only $\frac{3}{4}$ of the environment).
- The rate of crash and the rate of partial exploration is negatively correlated (correlation for the three environment = -0.91). Those who partially visit the environment are those who, by doing a wall following, miss the internal room. Thus it seems that if one follows walls it' is easier for it to avoid to crash.

First we will try to understand how the stochastic neuron with different values of k is used by the agents.

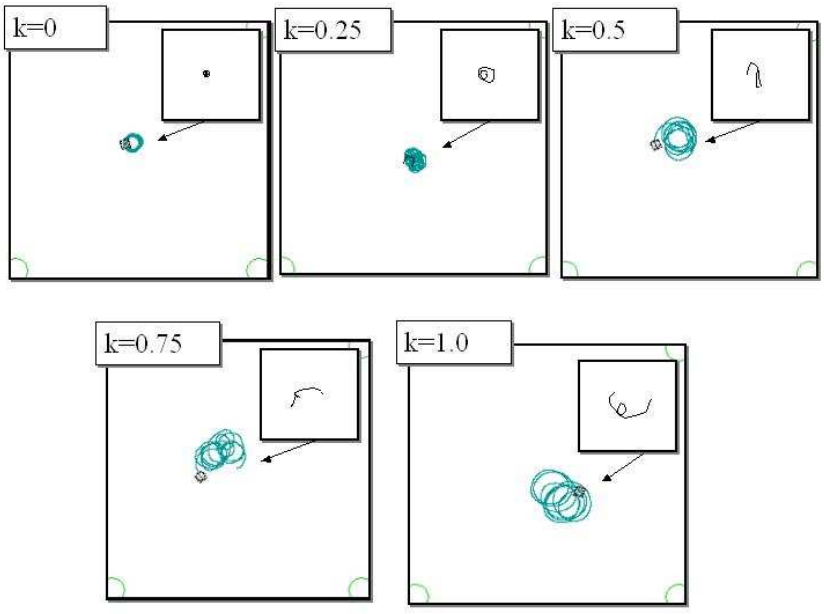


Figure 6.5: The 5 agents with values of $k = 0, 0.25, 0.5, 0.75$ and 1.0, the agent are tested for 5000 life steps on an ambient where walls are not visible from the starting point; (Up Right Squares) the variation of the center of the circular path that the agent is doing

The stochastic neuron gives some kind of noisy behavior to the agent (as could be seen of figure 6.6 Right), but its contribution becomes really important when the value of the infrared sensors are low, i.e when the agent is not near a wall and has few information coming from the environment. To illustrate this fact we test the five different agents in an ambient where the walls are not visible (see figure 6.5).
It is interesting to see that the space covered by the agents (in 5000 time steps) is proportional to their value of k. The simple reactive agent, turns always in exactly the same way, when the other agents have tendency to change the path. Notice that what we mean by "covered space" has nothing to do with the diameter of the circle, but with the variation of the center of this circle, as shown into the right up squares.

Now we can try to explain the difference of success rate in function of environment and the value of k by analyzing an example of test with $k = 0.75$. We will also explain why this agent shows good abilities to explore Environment 1, but not Environment 3.
Let consider the best individual of the experiment with $k = 0.75$ and a simple reactive agent on Environment 1. When the simple reactive agent is in a macro-cycle, it always follows exactly the same path, whereas a reactive

robot with a stochastic neuron has a more changing path even when it is in a cycle as shown in figure 6.6(Left).
If we compare the paths of the two different agents (figure 6.6(Right)), we could notice several differences. The most interesting is the third one (figure 6.6 (Right bottom)), that shows how the agent with stochastic neuron is able to visit once the left room and another time the right room.
The difference is about how the agent enters in the critical zone where it has the choice between more then one way. In Environment 1, in this particular position, the values of the other sensor inputs are near to 0, as there is no visible wall on right or left. The simple reactive agent turns always with the same angle (as we have seen on figure 6.5), even if we let the reactive agent live a long time(example 20'000 life step when it does a cycle in 4'000 life steps), nothing changes, it visits only three rooms, by doing exactly the same cycles. Instead the agent with the stochastic neuron, when confronted with the critical zone, has variable behavior, the angle of turning could be different from one to another passage.
Now that we have explained why the agent with stochastic neuron has a higher success rate than the normal reactive agent, we have to understand why its success rate is low when tested on Environment 3 (see figure 6.7)
The problem here is that when the agent is in the critical zone (bottom right) the values of the other inputs are not low, so the stochastic neuron cannot be used to visit both directions.
Notice also that in the corridors the agent's path is not perfectly rectilinear, because of the noisy behavior introduced by the stochastic neuron. This fact explains also why the agent with $k = 1$, is sometimes able to explore Environment 3. This agent has an even more noisy path into the corridors, and so, sometimes, when it arrives in the critical zone, it is far enough from the right wall to use the stochastic neuron and enter in the internal room. On the other hand, it is because of its very noisy behavior that its crash rate is so high.

This analysis could appear tiresome and very specific for one particular problem to the reader, but actually it's important to understand that introducing variability we are in confronted to a dilemma, if the neuron is completely random we have agent able to visit at least one time integrally any one of the three environment , but at same time we see the probability of crashing arise.
Instead a stochastic neuron with a lower level of random (example $k = 0.75, 0.5$) is useful for exploring a particular kind of environment: those in which the stochastic neurons which could be used in the critical zones. In both cases, it is not a *satisfying* strategy.

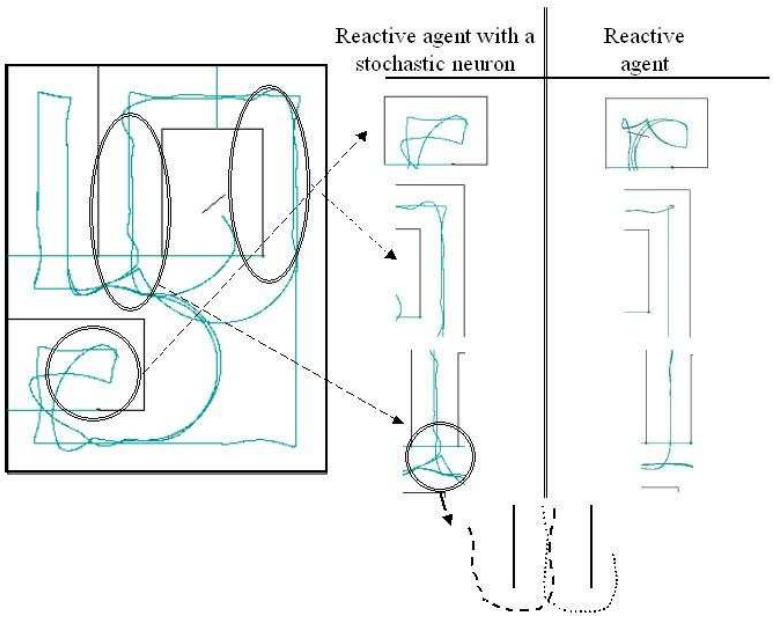


Figure 6.6: (Left) Path of a reactive agent with a stochastic neuron; (Right) Comparison between a simple reactive agent and a reactive agent with a stochastic neuron; bottom how the second model of agent manages to explore the internal room

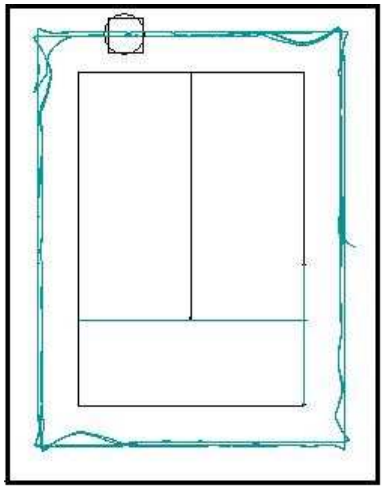


Figure 6.7: Path of a reactive agent with a stochastic neuron (k=0.75), tested on Environment 3, the variability of the agent's behavior is not enough to enter into the internal room.

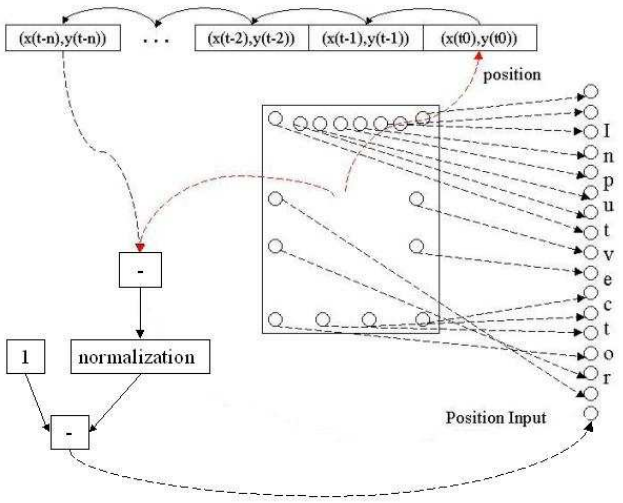


Figure 6.8: The Koala Robot and it's 16 infrared sensors, using odometry the robot calculates it's actual position that it a) Memorizes in the queue b) Compares with it's position at $time - n$ and injects it as a new input

6.4 External memory encoding the previous robot position

We introduce here for the first time, an egocentric information about the environment. This will be done by adding a queue[3] that retains the robots' relative position.
The implementation is schematized on figure 6.8. At every life step the position of the agent is registered in a queue of dimension n, and the difference between the actual position and the position at time $t - n$ (i.e the displacement) is calculated as :

$$Displacement[t_0] = normalized(|x[t_0] - x[t_0 - n]| + |y[t_0] - y[t_0 - n]|) \quad (6.6)$$

The input value is given by 1-*Displacement*. The rest of the neural architecture (not shown), is as for the precedent experiments, i.e. 4 hidden neurons and two output neurons.
The idea was to help the agent to get out of the cycles, by adding an input that had a value near of 1 if it had already been in the same position before (see figure 6.9). The most important parameter here is obviously n, that gives both the memory length and the determined past life step that is used to calculate the *Relative Position* input. We will see that a serious limitation

[3] a First In First Out (FIFO) list

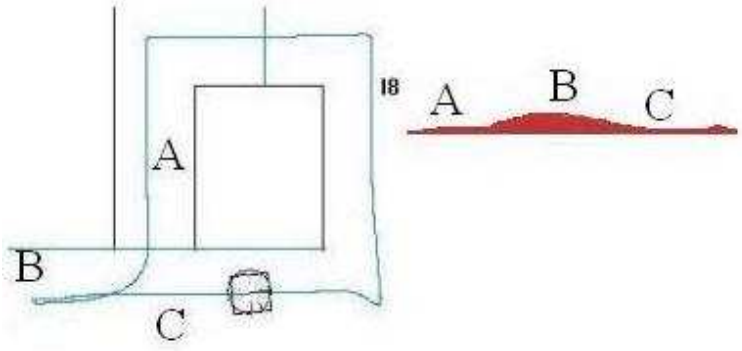


Figure 6.9: Simulated Robot and the corresponding activation for the input using the the difference between actual and $time - n$ position, here $n = 50$ life steps, B: the value of the input is near of 0 when the agent returns in a point where it was 50 time steps before

of this method resides in the fact that a good choice of n depends highly of the environment's shape and dimensions. If the agent is in a cycle, but $n <$ time needed for cover the cycle, the *Position Input* is not anymore useful.

It's important to notice that this method is not the same as the Dead Reckoning that implies the integration of all paths. Here we have only an information about a relative position in a particular instant in past. The cumulation of error is prevented by the fact that we calculate the position compare to a moving reference and not to an absolute point. Metaphorically said it is a Ariadne's thread with a finite length n dragging behind Theseus; one could notice that he is doing a cycle if the length of the cycle is shorter than the length of the thread.
This method could be easily used even with the Koala robot by using odometry and a magnetic compass (see section 2.3). The error rate is more or less always the same and we can control it with the value of n.
We will not present all the experiments with all different values of n that have been tested, instead will analyze two different ways to use this egocentric information: using a single value of n and one input neuron, or using a couple of value of n (typically one very high the other very low) and two corresponding neurons.
First of all, let consider an experiment with $n = 400$, i.e. with a memory that registers the 400 last positions of the the agent. In our case 400 life steps is a low quantity of memory. In fact, as it has been said, the entire environment could be explored a least in 4000 life steps. To understand how this new input changes the agent's action, we test it on an empty environment (figure 6.10 Left). The interest of the input appears clearly: when the agent begins to turn an makes a cycle, the value of the new input raise, and makes change

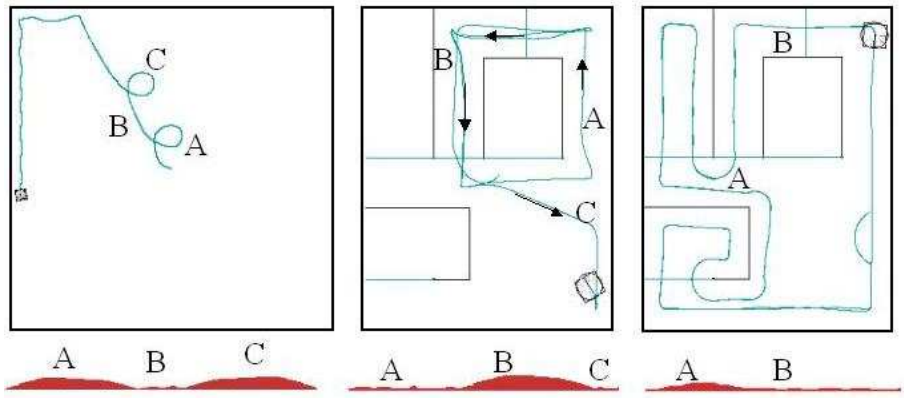


Figure 6.10: An agent with an input using egocentric information, with $n = 400$, Up (Left) the agent tested on a empty environment, (Center) The agent is able to get out from little cycles (Right) The agent is not able to get out of macro cycles; (Down) The corresponding values of the the input using the egocentric information

the agent's path. Then, as the agent is going in a new direction, and doesn't make any cycle anymore, the value of the input became null and the agent re-begins to cycle.
Now lets consider two cyclic paths of the agent (figure 6.10 Center and Right). The first one is smaller that what an agent could traverse in 400 time steps, when the second one is longer. Thus in second case the value of the neuron stays near to 0.
One could argue that an easy way to solve the problem is to have the biggest memory possible; for example if the environment needs 5000 life steps to be explored integrally, we could add have a memory of 5000 life steps. This "solution" should prevent any possible cycle. However as we know that the calculus of the position is not very precise we prefer to limit the cumulation of error. Thus the maximum value of n that we consider is $n = 2000$, during which, the agent could traverse approximatively 15 meters (the error will be $\pm 15cm$, see section 2.3).

The other solution explored is based on two neurons using the egocentric information, one with a low value of n ($n_1 = 300$) and the other with a high value of n ($n_2 = 2000$). The principal interest of this solution is that the vector of input, given by this two new inputs, is very variable. The agent has seldom two times exactly the same values from input with n_1 **and** input with n_2 as entry. This variability of input gives results near of what we have seen with the stochastic neuron, **the vector** $\begin{bmatrix} \text{input with } n = n_1 \\ \\ \text{input with } n = n_2 \end{bmatrix}$ has stochastic values. We illustrated this affirmation with figure 6.11.

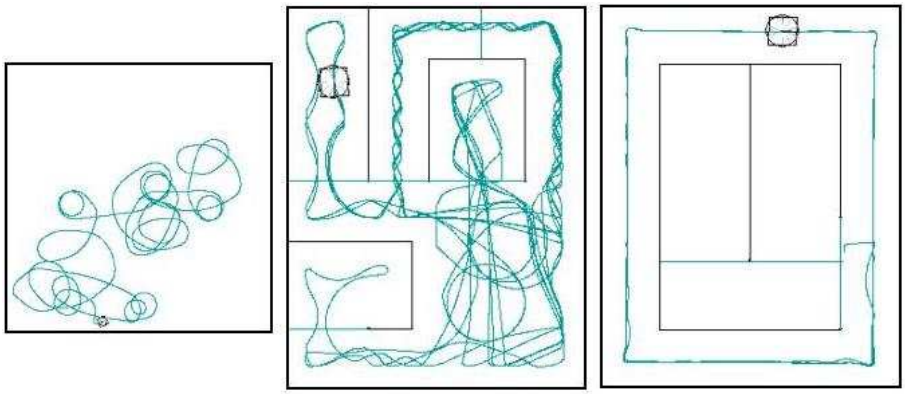


Figure 6.11: An agent with two inputs using egocentric information, with $n_1 = 300$ and $n_2 = 2000$,(Left) the agent tested on a empty environment shows a chaotic behavior, (Center) The agent has a very variable behavior, (Right) When the cycles is to long and the path traverse a very extended area of the environment, the variability disappears

The difference with the agent with the stochastic neuron is that, here the "random" behavior appears only if the agent stays in the same area of the environment. Instead if the agent is not cycling (or if it's cycles is longer than n_2) **and** if the it traverse a very extended area of the environment the random behavior disappears completely.
What makes this solution more interesting than the one using only one neuron, is that here even if the period of the cycles is longer that the "memory" of the agent, it's enough to have a cycles that is not extended in the environment to be able to go out the cycle. And what makes the method interesting compare to the solution with the stochastic neuron is that the random behavior disappear if the agent is correctly exploring the Environment, for example if the agent is going straight on in a long corridor, it's path will be rectilinear. This fact lowers also it's probability to crash as we could see on the next tabular.
The two type of agent (left part for the agent using one external memory, right part for agents using two external memories) tested on Environments 1, 2 and 3 (average of 20 experiments for each type of agent and each type of environment)

Completely visited	15	40
Crash	10	20
Partially visited	75	40

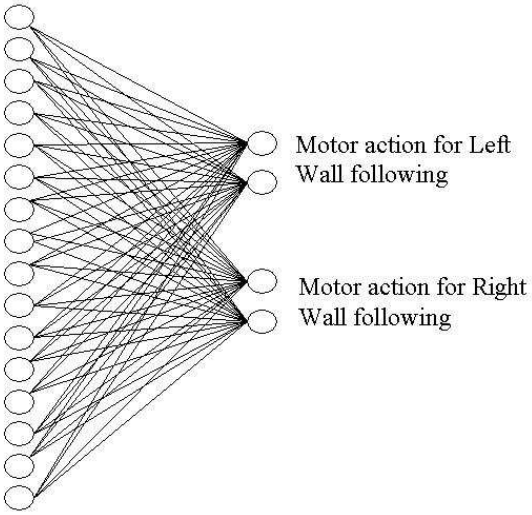


Figure 6.12: Modular architecture, with two pre-designed modules

6.5 Modular architectures

Till now we have seen specific type of agents that were sometimes able to visit some kind of environment but didn't use general strategy to explore all environments.
Instead a very general way to explore could be the use of an *accurate* sequence of Right and Left Wall Following.
For this issue we will introduce a modular architecture. In addition we will see a modelling for exploration in term of Right and Left wall following, finally we will try to define what is an *accurate* sequence. We will call a WF-strategy one of the two wall following (exclusively Right or Left). Thus exploration strategy will be a sequence of alternate WF-strategies.

A modular architecture can be used when we can subdivide a particular task in subtasks that are may be not compatible. This kind of architecture has been introduced and used successfully by S.Nolfi [24, 25].
The idea was to evolve two different neural subnetwork: one specialized for the left wall following, another for the Right wall following and a modula that chooses at each life step one of this two subnetworks. As shown on figure 6.12 the modular architecture is a duplicated neural network that permits to separately evolve two tasks. The choice of one or the other subnetwork is what we will call the *Sequence* of Right and Left Wall following.

6.5.1 Redefinition of exploration in term of Right and Left Wall following

We have given a definition of an environment as a graph, with nodes representing the rooms and edges representing doors (see chapter 4). However, if we want to define a strategy based on wall following, we should redefine our graph problem. A WF-Graph (Wall following Graph) is an undirect Graph $G(E, V, \Psi)$ where a node represents parts of the environment that could be visited with one particular wall following (exclusively Right or Left), and edges the position where the agent could change from one WF-strategy to the other. We give here this new formalization.
Any environment En could be expressed as a set of Cp, with Cp a set of points connected by segments.

$$(x, y) \in Cp_j \iff \exists (x_i, y_i)[(x_i, y_i) \in Cp_j \text{ and } (x_i, y_i), (x, y) \text{ are connected with a segment}] \tag{6.7}$$

$$Cp_j \subset En \iff \forall Cp_i \subset En, Cp_i \bigcap Cp_j = \emptyset \tag{6.8}$$

Let's consider the Environment 4 (figure 6.13), according to our definition 6.7 this environment has three sets of connected points, A, B and C. Now, an agent positioned near of one of this different set of point, for example A, will visit (or pass near of) all the points of A by doing exclusively a Left (respectively Right) Wall Following. If after certain time the agent changes its strategy and uses a Right (respectively Left) Wall Following, it will visit B, and so on by changing ones again it's strategy it will visit C.

We informally define with VCp_i the set of all points of the Environment that could be visited if one follows the walls defined by Cp_i. In fact as we can see on figure 6.13 an agent who follows for example A do not cross the set of points of A but VA.

Def 1 *VCp_i (vicinity) the set of all points of an Environment En that could be visited if one follows the walls defined by the set of connected points $Cp_i \subset En$*

We can constat that there are particular position where the robot can change from one WF-strategy to another. In fact if we consider ones again the figure 6.13, if the agent is in the right part of the environment and is doing a Right wall following it cannot change its strategy to a Left Wall Following because there is no visible wall on it's Left (as the right wall of C is not visible for the Koala's sensors). So there is no way to go directly from A to B.

We introduce here the definition of a set of points that connects two different sets of connected Points. The set of connection points $S_{CP}(C_i, C_j)$ is defined as:

$$(x,y) \in S_{CP}(Cp_i, Cp_j) \Longleftrightarrow \exists (x_i, y_i) \in Cp_i, \exists (x_j, y_j) \in Cp_j \\ [E_d((x,y),(x_i,y_i)) < M_s \text{ and } E_d((x,y),(x_j,y_j)) < M_s] \tag{6.9}$$

with E_d the Euclidean distance defined as $((x,y),(x_i,y_i)) = \sqrt{(x - x_i^2) + (y - y_i^2)}$ and M_s the maximum range of the Koala's sensors (i.e $\sim$ 20 cm).

Using the VCp notation, equation 6.9 is equivalent to :

$$S_{CP}(Cp_i, Cp_j) = VCp_i \cap VCp_j \tag{6.10}$$

We introduce the relation *is Robot Connected* (Rc) as following:
Two sets of connected points Cp_i and Cp_j are Robot Connected if there is at least one position in the environment where the robot can perceive at least one element of Cp_i and one element of Cp_j.

$$Rc(Cp_i, Cp_j) \Longleftrightarrow Cp_i \neq Cp_j \text{ and } S_{CP}(Cp_i, Cp_j) \neq \emptyset \tag{6.11}$$

Now we can formulate WF-graph as:

Def 2 *The WF-graph $G(E, V, \Psi)$ of an Environment En is an undirect graph where: E is the set of $Cp \subset En$, and there is an edge between two nodes Cp_iCp_j if $Rc(Cp_i, Cp_j)$.*

We have to redefine the set of all visitable environment E'_V in term of wall following as:

$$E \subset E'_V \Longleftrightarrow \forall Cp_i \in E, \exists Cp_j \in E[Cp_i \neq Cp_j \text{ and } Rc(Cp_i, Cp_j)] \tag{6.12}$$

Using WF-graph, that means that an environment is visitable if an only if its WF-graph is connected.
Notice that $E'_V \subset E_V$: the set of all environment that have been considerate at the very beginning of our research (see section 4). The set of environment that we don't consider anymore $\neg E'_V \bigcap E_V$ are environments that have internal rooms not visible for the Koala's sensors. Theoretically this set could be reduced as much as we want by augmenting the Robot's sensors range (by adding other type of sensors for example).
Returning to our example (figure 6.13), we have 3 set of Cp, and we have $Rc(A, B)$ and $Rc(B, C)$; with equation 6.12 we obtain that Environment 4

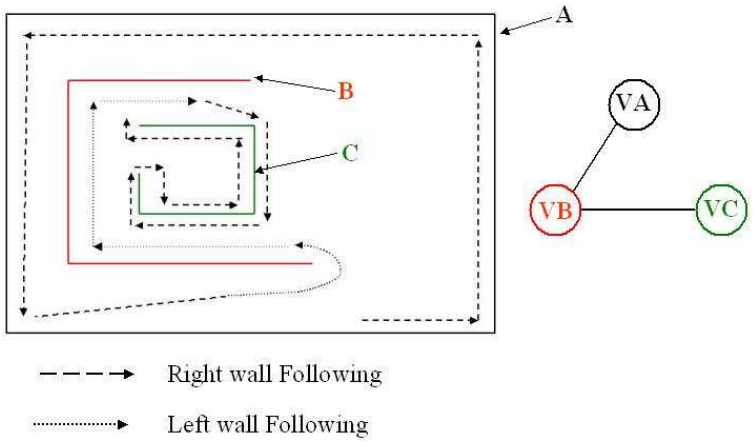


Figure 6.13: (Right) Environment 4: three set of connected points $A, BandC$, bold arrow: Right wall following, light arrow: Left wall following (Left) Corresponding WF-Graph

is a visitable environment(we should be able to visit it integrally by doing a sequence of Right and Left wall following). Reasoning with graphs we could have the same conclusion as WF-Graph of Environment 4 is connected.

The main problem here will be to define this sequence of changing strategy. We will show that even a random strategy (arbitrary changing of strategy on each life step) will give us a *acceptable* exploration, but obviously this method will be suboptimal in term of time and will not be *satisfying*.

6.5.2 Evolving a good wall following strategy

Thus the first part was to find a good fitness function and again a good environment, that could give us agents that follow for example the wall on right without regard to what is on it's left, i.e even if there is a wall very near on left, it will look for the wall on right.
As the wall following task is not very complicated in term of sensory motor coordination, the fitness, as well as the environment should be as simple as possible. My first attempts were with complicated environment (see figure 6.14 (Left)), or even changing environment (i.e. at each epoch the environment configuration changed). The fitness function was roughly implemented as follows: we defined an ordered list of big zone (numerated on figure 6.14), if an agent crossed two zone in the incremental direction, a good score S was given

In addition little zones were defined inside the big zones and if the agent crossed this zone in the good direction, a score $2 * S$ was given. The idea was to evolve agents that do a Left wall following, the agent had to follow the "C" form in the center and not follow the other walls. The little zone were the ideal position to the wall that we hoped to have, the big zone were defined to avoid the bootstrap problem (i.e. if we had only the little round zone the task could have been to difficult, and the fitness of all individuals of first generation could be null, avoiding any possible evolution, (p.13 in [12]). The evolved agents showed good abilities to follow the "C" but were to specific for this complicated environment, and tests with other environment didn't success, as they didn't had ability follow the Left wall but a specific shape.
At contrary, evolution with a very easy environment as defined on figure 6.14 (Right Up), gives much better solution. The fitness function was a particular form of obstacle avoidance:

$$\phi'' = (V)(1 - \sqrt{\Delta V})(1 - \lambda); \tag{6.13}$$

With V and ΔV given as in equation 6.2 and λ defined as:

$$\lambda = \begin{cases} 1.0, & \delta(\text{walls,agent}) = 300; \\ \text{linar function } f, & 300 < \delta(\text{walls,agent}) < 700; \\ 0.5, & \delta(\text{walls,agent}) = 700; \\ 0.1, & \text{else.} \end{cases} \tag{6.14}$$

Where δ gives the distance to the nearest wall.

The agents were evolved for 4 epoch, and at each epoch a different round zone was selected as start position. Their initial direction was set with regard to the selected position, to have the agent in the right direction for the corresponding wall following that we wanted to obtain.
It's interesting to notice that the second implementation permit to have agents able to manage with much more complicated environments (see figure 6.14 (Right Down)).

6.5.3 Evolve two separated Neural Networks

When a good strategy for an exclusive Right or Left wall following was obtained, we evolved agents with two separated neural network: first one evolved with Left wall following strategy and the second one with Right wall following strategy.
The architecture was as the one shown in figure 6.12 and the fitness function was ϕ''. The evolution was done as follows: first we evolved the agent for 300 generation on the ambient shown in figure 6.15 (Left). Every individual was evolved for 4000 life steps and 2 epochs. At the beginning of each epoch

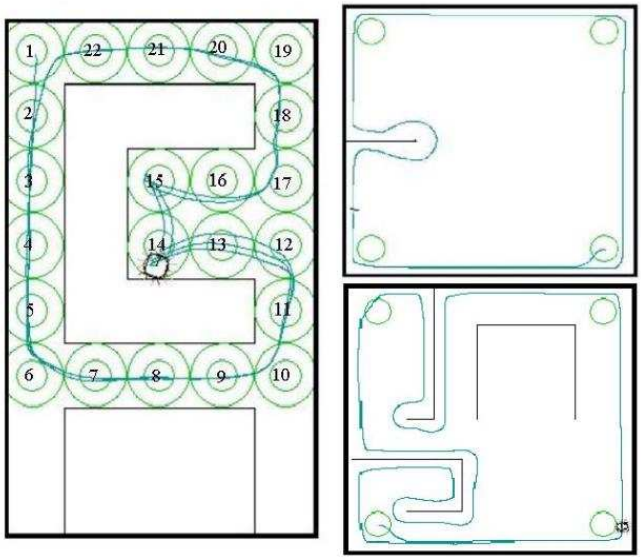


Figure 6.14: (Left)Environment 5: a complicated environment, not adapted for evolving wall following, round zones: if the agent cross these zone it has a positive score; (Right Up) Environment 6: a more easy and adapted environment, one round zone is randomly chosen as a start point at the begging of each epoch; (Right Down)Environment 7: Test on a more complicated environment success

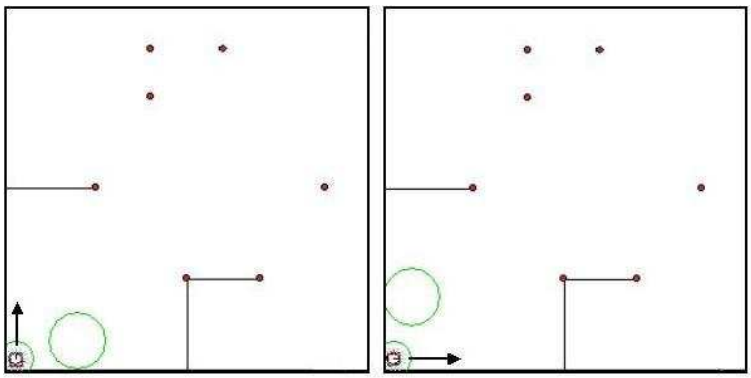


Figure 6.15: Environment 8, arrows gives the initial direction of the agent at the beginning of each epoch, big rounds are "holes", little rounds are obstacles; (Left) Environment for evolving a Left wall following (Right) Environment for evolving a Right wall following

the agent was positioned in the bottom right part of the environment and turned toward up. The big round on it's right is a "hole", the agent cannot perceive it but if it goes on the round, it crash. This was done to force the agent to make Left wall following. The little rounds were added to teach the agent to continue following the wall even it there were obstacles on right.
Then, when a good population was found, we re-evolve it on the second Environment (figure 6.15 (Right)). Here we evolve agents able to perform the Right wall following. During the re-evolution, mutation was done only on the second modula. In fact without this precaution, the agents could loose their ability to perform correctly the first task after the second evolution.

6.5.4 Define a sequence of Left and Right following

Here we will show first of all that the strategy is *acceptable* for any environment subset of E_V (set of all visitable environment defined in equation 6.12), if we use a random sequence of alternated WF-strategy. However, to render this strategy *satisfying*, we should define a good sequence.

Random sequence

We will consider again Environment 4. First we discretize our environment in zone that the agent crosses in n life step (see grey and black rounds in figure 6.16) and that we will call n-life zone, and assume that the agent could change WF-strategy every n-life zone, i.e one of this two possibility is selected randomly: continuing with the same WF-Strategy or change to the other WF-strategy.

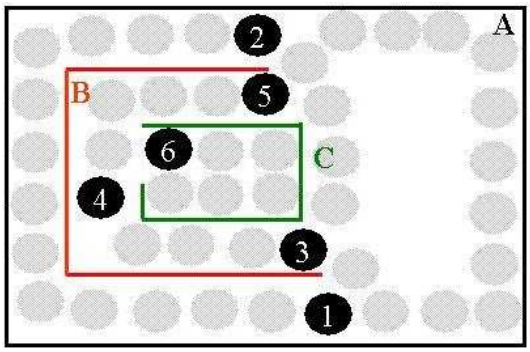


Figure 6.16: Environment 4, subdivision in zone that could be crossed in n-life steps: round gray zones; round black zones: border zones between two sub-zones of the environment.

We can divide our environment in a set of sub-zone, that are set of n-life zones:

Def 3

$$AB = S_{CP}(A, B)$$

$$A' = VA - AB$$

$$BC = S_{CP}(B, C)$$

$$B' = VB - BC$$

$$C' = VC - BC$$

with $S_{Cp}(Cp_i, Cp_j)$ a set of connection points between Cp_i, and Cp_j as defined in equation 6.10, VCp_i the set of vicinity points as defined in definition 1 and "$-$" the normal substraction operator as defined by theory of set.
The border points B_p (in black in figure 6.16) are n-life zones between two sub-zones, for example the border point 1 is between A' and AB.
Our exploration graph $G_E(E, V, \Psi)$ (see figure 6.19) will be defined as follows: nodes are border points and edge between two nodes means that there is a direct path between the corresponding border points, we give also the probability of chousing such a path (considering that at each n-life zone the agent chooses randomly between continuing it's WF-strategy or changes to the other) and it's length.
The *acceptability* of the exploration strategy is insured by showing that G_E is strongly connected, i.e. by finding a circuit that visits all node. $1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 5 \rightarrow 2 \rightarrow 1$ could be a possible circuit.
But the average time needed to explore an Environment cannot be calculated, as an agent could cycle indefinitely, for example $3 \rightarrow 4 \rightarrow 5 \rightarrow 3$.

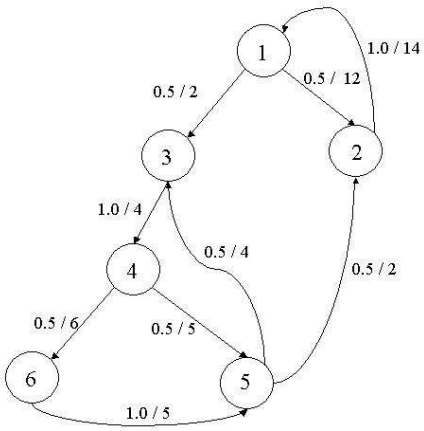


Figure 6.17: Direct Exploration Graph of Environment 4; nodes: border zone; edge between nodes a and b: path between a and b, label on edges: probability to chouse the corresponding path / length of the path.

In annex (see B)we give the demonstration that, for any Environment include in the set of visitable Environments E'_V, a random sequence of WF-strategy is an *acceptable* exploration strategy (by showing that the exploration Graph of an environment $En \subset E'_v$ i strongly connected) but not a *satisfying* one (by showing that an average time for visiting all rooms cannot by defined).

Fixed Time sequence

Another possibility could be to have a fixed time sequence. The idea was simply to change WF-strategy every fixed m life steps. As we will see this methods could give very good results and the exploration could be performed very quickly sometimes, but the problem is that, at contrary to the random sequence, here the acceptability is not insured.

For example, consider a strategy of fixed time sequence of alternated WF-strategies with $m = 13 \times n$ life steps on Environment 4 (see figure 6.16), with n the time needed to cross a gray (or a black) zone (the agent will change WF-strategy after having crossed 13 round zones).
If the agent begins on the the border zone 1 (black round), and do a Right wall following, it will continue till arriving in the gray round positioned on the Left of the border zone 2. Here it will change it's WF-strategy and will continue forward by following the Left wall, till arriving back to the border zone 1 where it will change again it WF-strategy, and so on it will cycle

indefinitely, and will never visit internal rooms B and C.

6.5.5 Results and critics

In this section we will present the results for the two different type of sequence of WF-strategy that we have seen. The best individuals of the best seed for both method have been tested on Environment 1,2,3. On figure 6.18 we show the average percent of complete exploration compared to partial exploration, in 20'000 life steps.
The table on right gives the average time (in thousand life steps) for an integral exploration, and the square root of the variance. The test confirms what we have presented theoretically in the last section:

- With the fixed time sequence the probability to cycle is much higher. As we could see the rate of total exploration is much lower, particulary for low values of m. The problem is, as we explained, that the agent could indefinitely cycle. Instead using a random sequence, there is always a possibility to explore integrally.

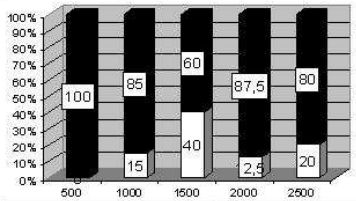
- However, for those that don't cycle, the performance are better than for the random sequence. The average time for exploration is much lower.

- The square root of variance of the the random sequence is very high. That confirms what we have theoretically shown, average time for the random sequence is not insured.

- The value of n is negatively correlated with the success rate, for the random sequence. That could be understood as follows. If the gray zone (that are defined by the value of n) on figure 6.13 are very big we could have the same problem as for the fixed time sequence, i.e. the agent could indefinitely cycle.

To illustrated the relevance of the method using a random sequence, we give here an example of a a successful test for each Environment. As we could see agents using this method shows good ability to explore even complex environment. The only problem that we haven't resolved yet is that sometimes the agent doesn't change wall when it changes WF-strategy, but simply turn and continue in in the other direction. To avoid this problem, that decrease performance in term of time, we have tried serval possible method: revolving agents in a very big environment, to force them to go straight when the sensory inputs are low or use a third modula that is activate between two WF-strategy and that is evolved for an obstacle avoidance, none succussed yet.
In addition the crash rate is for the moment high: $\sim 15\%$ indifferently for the two methods and the different ambient. We haven't include it in the

m	500	1000	1500	2000	2500
average time	0	9,5	8	7	5,5
square root of variance	0	0,92	1,29	0,61	0,71

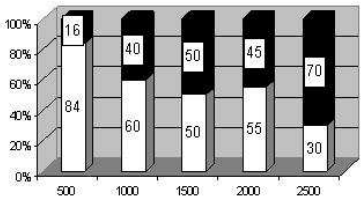


Fixed time sequence

Partially visited

Completely visited

m	500	1000	1500	2000	2500
average time	12,5	11	9	10,5	8
square root of variance	5,72	3,39	2,28	2,77	3,79



Random sequence

Figure 6.18: (Left) Average percent of complete and partial exploration in 20'000 life steps on Environments 1,2 and 3 with the two types of sequences of WF-strategy proposed, (Right) Average and the square root of the time (in thousand life steps) for tests that have succussed

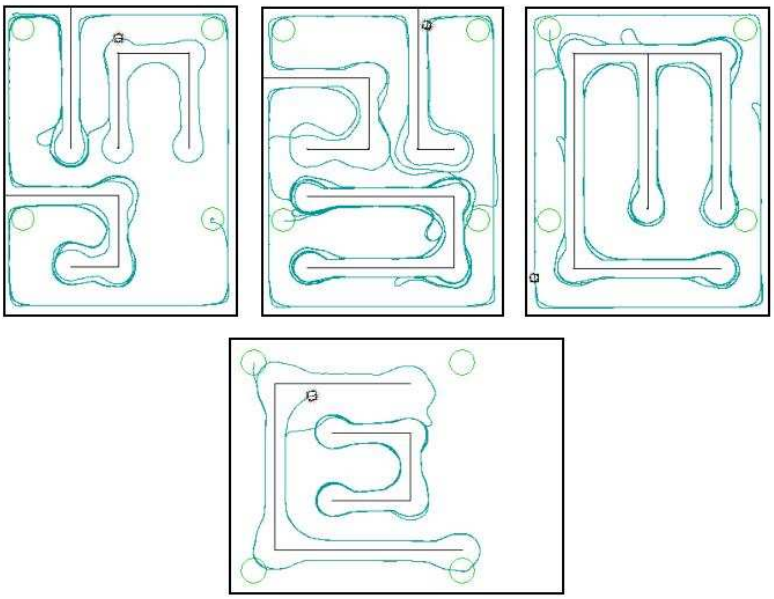


Figure 6.19: Agent using a random sequence of WF-strategies, $n = 1000$ life steps, form top Left to Right: Environment 1,2 and 3, down: Environment 4

statistics because it has nothing to do with the to types of strategy (nor with the value of m), it is simply a consequence of a problem that occurs in the precise moment in which the agent changes from one WF-strategy to the other: if the agent is in a corner, it crashes.

Chapter 7

Conclusion

Our research begun with the the human target following task, and we realize that if a Robot had to follow a human, it had also to manage exploration when it loses its target.

We will discuss here our results in basis of what were our initial statements: *Defining a robust strategy, using evolutionary robotics methods, to perform the exploration task with (Hybrid) reactive agents with two constraints: Space (The robot should be able to explore every part of the environment) and Time (It has to do it as fast as possible, avoiding as much as possible cycles).*

Looking for particularities of realistic environment, we discovered the limitation of the simple reactive agents, and showed that they were unable to explore environments that comport internal rooms (that is correlated with the aliasing problem).
Thus we have introduced three hybrid reactive agents: A reactive agent with a stochastic neuron, A reactive agent with an external memory encoding its precedent actions, An agent with a modular architecture.
The conclusions we made are the following:

a A possibility to explore, and go out of cycles is to add some randomness on the agent's behavior. This method confronted us with a dilemma: if the behavior was very random, the agent was sometimes able to explore correctly but on the other hand its probability to crash increased. Istead, a lower level of randomes did'n't permit reactive agents to explore and go out of cycles.

b Another possibility was to add an information about the environment. We used an external memory coding the previous positions of the agent. Here the problem was the length of the memory, short memomry prevented only few cycles, and long memory introduced errors because of the imprecise calculus of the postion. We have presented

also another experiment using two external memories with different length and we found agents which behavior was similar to agents with stochastic neuron. In fact by having two neurons using two external memories we augmented the variability of the agent in a intelligent way: the behavior became random when the agent returned in a place where it had been before or remained in the same area of the environment.

c Finally we used a duplicated feedforward architecture with a modula that chose at every life step which network was used. The idea here was to traduce the indoor exploration as a sequence of Left and Right wall following and evolve a distinct neural network for each task. The most challenging problem was to define how the modula had to choose the networks to be used. We have given in this project two very easy solution that still give good results: the modula chose randomly every m life step one of the two networks, or the two networks were alternatively selected every m life steps.

Our constraints were space and time. With all this new hybrid reactive agent we have considerably augmented the performence in term of space: the new agent are in general able to explore more parts of the environment than the simple reactive. The most interesting was the one using the modular architecture with a rapid random sequence of Right and Left Wall Following. However the time needed to explore in this case wasn't satisfying the agent could do a lot of cycles before exploring all the environment.

Future Research

We obtained the most interesting results by using an agent with a modular architecture. One of our limitation was the high rate of crash. We think that the problem is not inherent to the method, we should be able to avoid the crash, changing may be the environment on which the agents are evolved to perform the wall following. Our problem was that the agent learned how to follow wall on one side but didn't learn to avoid the wall on the other side.

Another possible future work could be to introduce internal states. I think that cupelling the modular method based on a sequence of Left and Right wall following with a pro-active architecture able to perform the self-localization task could give interesting results. The self-localization ability could be used to choose the right moment to change from one network to the other.

Bibliography

[1] S. Nolfi.(2000) *Evorobot 1.1 user manual.* http://gral.ip.rm.cnr.it/evorobot/simulator.html.

[2] K-Team S.A. Kephera robot(1998). *Koala Version1.1 User manual.* http://www.k-team.com.

[3] Miglino O., Lund H. and Nolfi S.(1995). *Evolving mobile robots in simualting and real environments*, Artificial Life, 2:417-434.

[4] Searle J.(1987). *Minds and Brains without Programs* , C. Blakemore and S. Greenfield (eds.), Mindwaves (Oxford: Blackwell)

[5] Varela F.J., Rosch E., and Thompson E. (1991). *The Embodied Mind: Cognitive Science and Human Experience*, Cambridge, MA: MIT Press/Bradford Books.

[6] Cressant A.(1999) *Étude in vivo des cellules de lieu, impliquées dans la mémoire spatiale*, thesis repport

[7] Beer R.D. (1995). *A dynamical systems perspective on agent-environment interaction*, Artificial Intelligence, 72:173-215.

[8] Benhamou S., Sauvé J.P. and Bovet, P. (1990). *Spatial memory in large scale movements: Efficiency and limitations of the egocentric coding process. J. Theor. Biol.*, 145, 1-12.

[9] Olton D.S. (1982). *Spatially organized behaviors of animal : behavioral and neurological studies.* In : Spatial abilities. Developmental and physiological foundations, M. Potegal, Ed., New York : Academic Press, 335-360.

[10] Beer R.D.(2001) *The dynamics of active categorical perception in an evolved model agent.* Behavioural and Brain Sciences, submitted.

[11] Braitenberg V. (1984). *Vehicles: Experiments in Synthetic Psychology.* MIT Press, Cambridge, Massachussets.

[12] Nolfi S. & Floreano D. (2000). *Evolutionary Robotics, The biology, Intelligence, and Technology of Self- Organizing Machines*, London, The MIT Press/Bradford Books.

[13] Whitehead S.D. and Bellard D.H. (1991), *Learing to perceive and act by trial and error*, Machine Learning, 7:45-83

[14] O'Keefe J. and Nadel L. (1978). *The hippocampus as a cognitive map.* Oxford, U.K.: Oxford University Press. Workshop: Connectionist Representation. London: Springer Verlag

[15] Floreano D. and Mondada F. (1994), *Automatic creation of an autonomus agent: genetic evolution of a neural-network driven robot*, In D.Cliff,P.Husbands,J.Meyer and S.W. Wilson(Eds.), From Animals to Animats 3: Proceedings of Third Conference on Simulation of Adaptative Behavior. Combrige, MA:MIT Press/Bradford Books.

[16] Bianco R., Caretti Massimiliano, Nolfi S.,(2003) *Developing a Robot able to Follow a Human Target in a Domestic Environment*,RoboCare: Proceedings of the First RoboCare Workshop. Rome: Edited by Amedeo Cesta.

[17] Miglino O., Denaro D., Tascini G., Parisini D.,(1998) *Detour Behavior in Evolving Robots: Are Internal Representations Necessary?*,EvoRobots 1998: 59-70

[18] Etienne A. S., Joris-Lambert S., Reverdin B. and Téroni E. (1993). *Learning to recalibrate the role of dead reckoning and visual cues in spatial navigation.* Anim. Learn. Behav., 21, 266-280.

[19] McNaughton B.L., Chen L.L. and Markus E.J. (1991). *"Dead reckoning", landmark learning, and the sense of direction: A neurophysiological and computational hypothesis.* J. Cog. Neurosci., 3, 190-202.

[20] Nolfi S.,(2002) *Evolving Robots able to Self-localize in the Environment: The Importance of Viewing Cognition as the Result of Processes Occurring at Different Time Scales*, Connection Science (14) 3:231-244.

[21] De Croon G.,(2004) *Pro-active agents with recurrent neural controllers*, thesis repport

[22] Hill B., Best P. (1981) *Effects of deafness and blindness on the spatial correlates of hippocampal unit activity in the rat.* Exp. Neurol., 74, 204-217.

[23] Bahadori S., Iocchi L., Scozzafava L., (2003) *People and Robot Localization and Tracking through a Fixed Stereo Vision System*, RoboCare: Proceedings of the First RoboCare Workshop. Rome: Edited by Amedeo Cesta.

[24] Nolfi S. (1997) *Using emergent modularity to develop control system for mobile robots*Adaptative Beahvior, 3-4:343-364.

[25] Nolfi S. (1997) *Evolving not trivial behavior on autonomous robots: A garbage collecting Robot* Robotics and Autonomous System, 22:187-198.

[26] Caprari G., Arras K.O. and Siegwart R. (2001) *Robot Navigation in Centimeter Range Labyrinths*. In Proceedings of the First International Conference on Autonomous Minirobots for Research and Edutainment, Heinz Nixdorf Institute, Paderborn, Germany.

[27] Johan Bos, Ewan Klein, and Tetsushi Oka. (2003) *Meaningful conversation with a mobile robot*. In Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL10), pages 71-74, Budapest

[28] Hartmann G. and Wehner R. (1995). *The ants path integration system: a neural architecture.* Biological cybernetics, 73:483-497

[29] Wehner R. Michel B. and Antonsen P. (1996). *visual navigation in insects: Coupling of egocentric and geocentric information*, The journal of experimental Biology, 199:129-140

[30] Kim D., Hallam J. C. T. (2000). *Neural network approach to path integration for homing navigation.* In Meyer J. A., Berthoz A., Floreano D., Roitblat H., S. W. Wilson (Eds.), From Animals to Animats 6 (pp. 228-235). : MIT Press.

[31] Poucet B., Benhamou S. (1997). *The neuropsychology of the spatial cognition in the rat*. Critic. Rev. Neurobiol., 1 (2&3), 101-120.

Thanks

This thesis would not have been realized in one human life without the great help of my Supervisors Stefano Nolfi at the CNR in Rome and Dario Floreano at the EPFL in Lausanne. I would like to thank those how have helped me during these four month: R. Bianco, for his innumerable hint and suggestions, Francesco Mondada for the precious help during my last chaotic days of work, but also my colleague in CNR Gianluca Baldasarre, Marco, Luca, Davide, Massimiliano and Emiliano.
Finally I would like to tanks Petra Krausz and Omar Shokur, for their help during the difficult writing process.
A last special thanks for Romulus and Remus for having founded a city like Rome.

Appendix A

Graphs

Direct and Undirect Graph

A finite Direct (respectively Undirect) Graph $G = (V, E, \Psi)$ is the triplet defined by:

- A finite set V ($|V| = n$) who's elements are called nodes
- A finite set E ($|E| = m$) who's elements are edges
- A function $\Psi : E \longrightarrow V$ x V, called function of incidence who associate for all edge $e \in E$ an ordinate (respectively non ordinate) couple $(u(e), v(e))$ of nodes, where the node $u(e)$ is the called initial extremity of the edge e (respectively adjacent with e) and $v(e)$ is the final extremity of e (respectively adjacent with e).

Chain and Cycle

A chain is an alternated sequence of nodes and edges $C = (u_0, f_1, u_1, f_2, u_2, ..., u_{k-1}, f_k, u_k)$ where $\forall i\ u_i \in V$, $f_i \in E$ and $f_i = \{u_{i-1}, u_i\}$

- A cycle is chain where the two extremity are the same node.
- A chain (or a cycle) is elementary if any node appear only once.
- A chain (or a cycle) is elementary if any edge appear only once.
- An undirect Graph is is acyclic if it no simple cycles.

Path and circuit

A path is an alternated sequence of nodes and edges $C = (u_0, f_1, u_1, f_2, u_2, ..., u_{k-1}, f_k, u_k)$ where $\forall i\ u_i \in V$ and $f_i = (u_{i-1}, u_i) \in E$.

- A circuit is a path where the two extremity are the same node.
- A direct graph is acyclic if it has no circuit.

Connectivity

Consider $G = (V, E, \psi)$ an **undirect** graph, we define on V a relation of **connectivity** C as:

$$v_i C v_j = \begin{cases} v_i = v_j \text{ or} \\ \text{There is a } \textbf{chain} \text{ between } v_i \text{ and } v_j \end{cases} \tag{A.1}$$

C is an equivalence relation: reflexive, symmetric and transitive. Classes of equivalence of C define the connected components of G.

A graph G is connected $\Longleftrightarrow$ G is composed by only one connected component (A.2)

Strong Connectivity

Consider $G = (V, E, \psi)$ an **direct** graph, we define on V a relation of **strong connectivity** CF as:

$$v_i CF v_j = \begin{cases} v_i = v_j \text{ or} \\ \text{There is a } \textbf{path} \text{ between } v_i \text{ and } v_j, \textbf{ and a path} \text{ between } v_j \text{ and } v_i. \end{cases} \tag{A.3}$$

CF is an equivalence relation: reflexive, symmetric and transitive. Classes of equivalence of CF define the strong connected components of G.

A graph G is Strongly connected $\Longleftrightarrow$
G is composed by only one strong connected component

(A.4)

Appendix B

Demonstration

Acceptable strategy

We give here the demonstration that a random sequence of Left and Right wall following is an acceptable strategy of exploration, for any environment in E'_V.
We have to show that: $E_n \subset E'_V \Longrightarrow$ the exploration graph $G_E(E, V, \Psi)$ is strongly connected.

0. $E_n \subset E'_V$
Justification: Hypothesis

1. $En = \{Cp_1 \cup Cp_2 \cup ... \cup Cp_n\}$ and $\forall Cp_i, Cp_j : Cp_i \cap Cp_j = \emptyset$
J: 6.8

1.1 The set of points that could be visited by following elements of En, $VEn = \{VCp_1 \cup VCp_2 \cup ... \cup VCp_n\}$
J: 1. and Def 1

2. $\forall Cp_i \in En, \exists Cp_j \in En[Cp_i \neq Cp_j$ and $Rc(Cp_i, Cp_j)]$
J: 0. and 6.12

3. $\forall Cp_i \in En, \exists Cp_j \in En[Cp_i \neq Cp_j$ and $(S_{CP}(Cp_i, Cp_j) \neq \emptyset)]$
J: 2. and 6.11

4. $S_{CP}(Cp_i, Cp_j) \subseteq VCp_i$
J: Definition 6.10

5. $S_{CP}(Cp_i, Cp_j) \Longrightarrow VCp_i = \Big(VCp_i \cap \neg S_{CP}(Cp_i, Cp_j)\Big) \cup S_{CP}(Cp_i, Cp_j)$
J: 4. and theory of set's property : $A = (A \cap \neg B) \cup B$ if $B \subseteq A$

6. $VEn = \Big\{\Big((VCp_1 \cap \neg S_{CP}(Cp_1, Cp_i)) \cup S_{CP}(Cp_1, Cp_i)\Big) \bigcup \Big((VCp_2 \cap \neg S_{CP}(Cp_2, Cp_j)) \cup S_{CP}(Cp_2, Cp_i)\Big) \bigcup \ldots \bigcup \Big((VCp_n \cap \neg S_{CP}(Cp_n, Cp_k)) \cup S_{CP}(Cp_n, Cp_k)\Big)\Big\}$
J: 1. and 5.

7. $VEn = \Big\{C'_1 \cup C'_2 \cup \ldots \cup C'_n \cup C_1C_i \cup C_2C_j \cup \ldots \cup C_nC_k\Big\}$
J: Def 3

Now we can continue the reasoning with graphs: we know that the WF-graph of En is connected (by definition of Environments $\subset E'_V$). In addition with proposition 7. we have a traduction of the environment VEn in sub-zones. The border points that defines the nodes of the exploration graph are simply the points between two sub-zones. We can show that the exploration graph is strongly connected if the WF-graph is connected, by passing by an intermediary graph using the sub-zones as shown on figure B.1.
We first traduce the undirect WF-Graph in a direct WF-Graph, by simply duplicating any edge e in e' and e''. The non ordinate couple of nodes (u(e),v(e)) associated to e will be traduced in two ordinates couples of nodes $((u(e), v(e))$ and $(v(e), u(e)))$ associated to e' and e''. The corresponding direct graph is strongly connected.
If the directed WF-graph is strongly connected the sub-zone graph is strongly connected, and by so the exploration graph is strongly connected, as the nodes are the points between two sub zone and edges are path between two border zone. The existence of the path is insured by the fact that we have based the exploration graph on sub-zones graph, path between two border nodes is the corresponding node of the sub-zone graph, for example the path between the border node i and j is $C_{pi}C_{pj}$.

Thus the exploration graph, for the exploration strategy based on a random sequence of Right and Left wall following is strongly connected.

Non satisfying strategy

We want to show here that an average time for exploring an environment integrally by using a random sequence of Right and Left following is not insured, and by so that the strategy is not satisfying.

The average time for exploring could be found by calculating the average length of the Exploration graph (see figure 6.19).
The average length is the average of the Expected Value of all the possible paths:

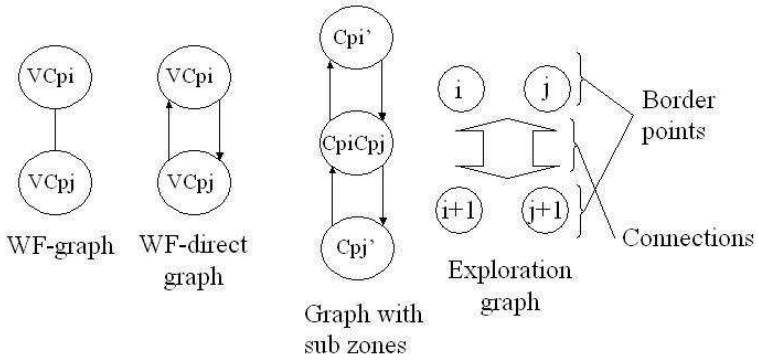


Figure B.1:

$$\frac{1}{n}\sum_{i=0}^{n} E(Path_i)$$

with n the number of possible path and $E(path_i)$ given by:

$$E(path_i) = \frac{1}{m}\sum_{j=0}^{m} \lambda(edge_j) \times P(edge_j)$$

with m the number of edges for the corresponding path, $\lambda(edge)$ and $P(edge)$ respectively the length and the probability of the the edge (as defined in our exploration graph on figure 6.19).
As our graph is cyclic, the number of possible path $n \to \infty$, and thus the average length is undefined: the strategy is not satisfying.