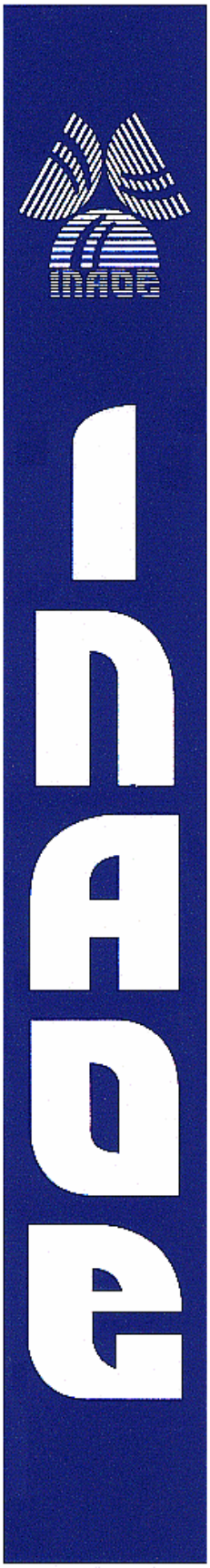


Development of Local Vision-based Behaviors for a Robotic Soccer Player

Por

Antonio Salim Maza

Tesis sometida como requisito parcial para obtener el grado de **Maestro en Ciencias** en la especialidad de **Ciencias Computacionales** en el Instituto Nacional de Astrofísica, Óptica y Electrónica.

Supervisada por:

Dr. Olac Fuentes Chávez
Instituto Nacional de Astrofísica, Óptica y Electrónica

Dr. Angélica Muñoz Meléndez
Instituto Nacional de Astrofísica, Óptica y Electrónica

Tonantzintla, Pue.
Septiembre 2004

Development of Local Vision-based Behaviors for a Robotic Soccer Player

by

Antonio Salim Maza

Submitted in partial fulfillment of the requirements for the degree of **Master in Science** in **Computer Science** at National Institute of Astrophysics, Optics and Electronics

Advisors:

Olac Fuentes Chávez, PhD

Angélica Muñoz Meléndez, PhD

September 3, 2004

To my parents Manuel and Yolanda, my brother Manuel, my girlfriend María Isabel and my wonderful family.

Abstract

This research focuses on the development of local vision-based behaviors for the robotic soccer domain. The behaviors, which include *finding ball, approaching ball, finding goal, approaching goal, shooting* and *avoiding*, were designed and implemented using a hierarchical control system. The *avoiding* behavior was learned using the C4.5 rule induction algorithm, the rest of the behaviors were programmed by hand. The object detection system is able to detect the objects of interest at a frame rate of 17 images per second. We compare three pixel classification techniques; one technique is based on linear color thresholds, another is based on logical AND operations and the last one is based on the Artificial Life paradigm. Experimental results obtained with a Pioneer 2-DX robot equipped with a single camera playing on an enclosed soccer field with forward role indicate that the robot operates successfully, scoring goals in 90% of the trials.

Resumen

Esta investigación se enfoca en el desarrollo de comportamientos basados en visión local para el dominio del fútbol robótico. Los comportamientos, los cuales incluyen: *buscar la pelota, aproximarse a la pelota, buscar la portería, centrar la portería, disparo a gol* y *evadir obstáculos*, fueron diseñados e implementados utilizando un sistema de control jerárquico. El comportamiento *evadir obstáculos* fue aprendido usando el algoritmo de inducción de reglas C4.5; el resto de los comportamientos fueron programados manualmente. El sistema de detección de objetos es capaz de detectar los objetos de interés a una tasa de 17 imágenes por segundo. Comparamos tres técnicas para clasificación de píxeles; la primera está basada en umbrales de color lineales, otra está basada en operaciones lógicas AND y la última está basada en el paradigma de vida artificial. Los resultados experimentales usando un robot Pioneer 2-DX equipado con una cámara, jugando en un campo cerrado de fútbol indican que el robot opera exitosamente, anotando goles en 90% de los intentos.

Acknowledgements

I would like to thank Olac Fuentes and Angélica Muñoz, my advisors, for their suggestions, corrections and constant support during this research.

I also want to thank my thesis committee. Rodrigo Montúfar expressed his interest in my work and supplied me with useful information of some recent work. Ariel Carrasco, Aurelio López and Carlos Alberto Reyes for their interest in my work and their valuable suggestions. I also like to thank CONACyT for the financial support received under grant number 171617.

My most special gratitude is to my parents, Manuel and Yolanda, for their patience and love. Also, I wish to thank my brother Manuel for his unconditional help and company. The most important acknowledge goes for María Isabel for her support, patience and company.

I cannot forget my friends I meet during my studies: Ignacio Algredo, Luis Álvarez, Carlos Avendaño, Tomas Balderas, Hector Barrón, Alejandro del Castillo, Edith Cuellar, Jorge Cruz, Carlos Díaz, Trilce Procyon Estrada, Topiltzin Flores, Carlos Arturo Hernández, Erika Danaé López, Luis David López, José Luis Saul Malagón, Miguel Morales, Liz Nicandi, Moises Pérez, Erick Rodríguez, Santos Martín López, Gerardo Sosa and Alberto Téllez.

Finally, I am grateful to Norma Tecampo, Lucia Angélica de la Vega and Nidia for their support along my thesis.

Puebla, Mexico — Antonio Salim
September 3, 2004

Table of Contents

Chapter 1

Introduction

1.1 Robotic soccer

RoboCup is an international competition for robotic soccer teams. RoboCup has been held every year since 1997. RoboCup is a common task for artificial intelligence and robotics research, which provides a test platform for evaluating various theories, algorithms and agent architectures [5]. RoboCup is an attempt to promote research using a common domain, robotic soccer. For RoboCup the ultimate goal according to Asada and Kitano [5] is: "By the mid 21st century, a team of autonomous humanoid robots shall beat the human World Cup champion team under the official regulations of FIFA". RoboCup consists of three general competitions [20]: the real robot competition, the software robot competition and the special skills competition.

Programming a robotic soccer player is an extremely difficult task for the following reasons:

- The game environment where the robot operates is highly dynamic, thus the robot must respond rapidly to changes on the soccer field.
- The integration of a broad range of technologies may be necessary. Some research areas are: agent architecture, combination of reactive and deliberative approaches, real-time recognition, reasoning and planning; sensor fusion, behavioral learning, strategy acquisition, cognitive modelling, vision and many more.
- Try to program by hand a robotic soccer player is difficult because the state-space is sufficiently huge to consider all possible situations.

- Robots must deal with real world complexities through its sensors and actuators. The main problem for robots is that sensor readings are noisy and actuators are inaccurate.

- Communication among players is limited, therefore, each agent should behave flexibly and must be autonomous.

- Designers must organize several subsystems such as vision, learning, control and reasoning in a controller, which runs all subsystems concurrently.

- Because of the limited perception gathered by sensors, there are similar world states where the robot must respond in different ways.

The experience obtained working on robotic soccer domain can be applied in a variety of applications, such as: planet exploration, cooperative rescue robots, cooperative map constructing, or another application which requires several cooperative robots to perform a mission successfully.

The RoboCup ultimate goal is very ambitious, thus this research is only a little step towards accomplishing the RoboCup ultimate goal. Our research focuses on designing and evaluating perceptual and behavioral control methods for the RoboCup Physical Agent Challenge [6]. Our incremental method for constructing a robotic soccer team is based on the design of a group of specialized robots in a particular task, for example: a robotic soccer player with forward role, a defence or a goalkeeper, thus we must design and implement a top-level coordination mechanism to control them as an intelligent team.

1.2 Global vision and local vision

Vision is the primary sense used by robots in RoboCup for image acquisition to identify the robots, the goals, the ball and their positions on the field. When designing the robots, researchers have two different types of vision systems available: global vision and local vision.

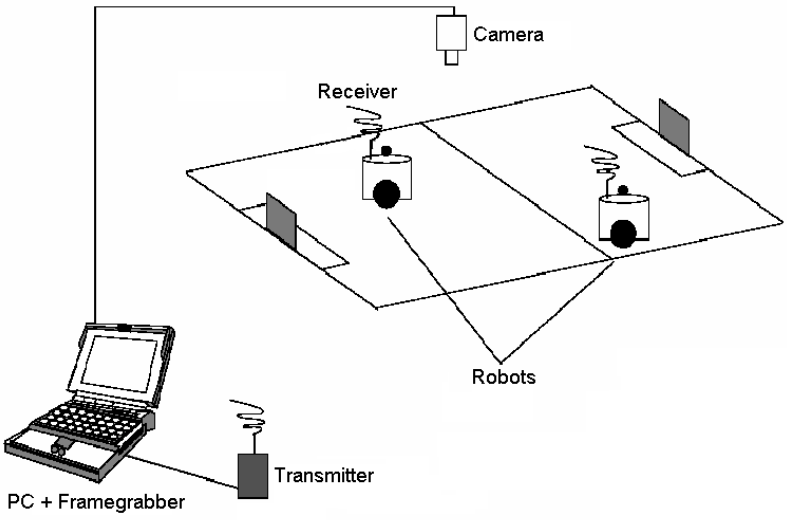


Figure 1.1: Global vision.

In global vision, a camera is mounted over the field. The image captured by the camera then is passed to an external computer that processes it and determines the commands for the robot. Figure 1.1 shows a generic diagram of the global vision approach.

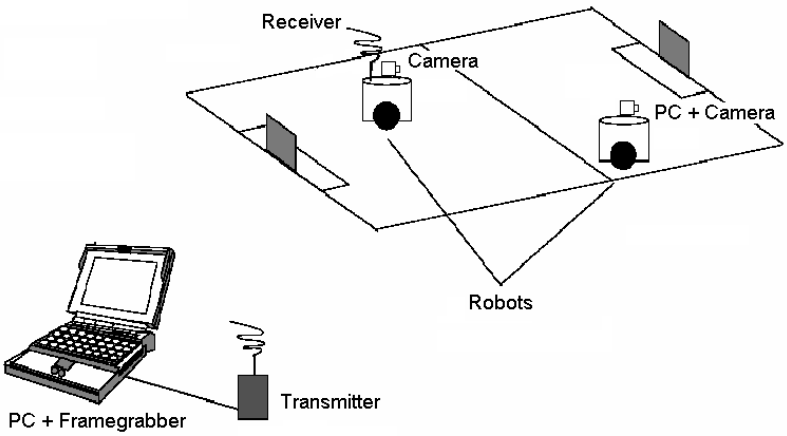


Figure 1.2: Local vision.

In local vision, the robot is equipped with a camera and the on-board or off-board image processing system determines the commands for the robot. Figure 1.2 shows a generic diagram of the local vision approach.

Global vision	**Local vision**
Only one image processing system is required to detect all objects of interest	Each robot needs an image processing system to detect all objects of interest
Overall vision field	Locally limited vision field
Robots know the exact position in the field of all objects of interest, there is not uncertainty	Robots have a partial perception of the world, there is uncertainty
Communication is sometimes necessary in order to take team decisions	Communication among robots is very necessary in order to take team decisions
Robots can focus on other issues such as control	Robots need to focus on perception and in other issues
This approach is focused on the development of distributed control	This approach is focused on the development of autonomous robots
There are not duplication of functions, the overall system is used to control all the robots in a team	There are duplication of functions for each robot, such as image processing and control

Table 1.1: Comparison between global and local vision

Table 1.1 compares global vision and local vision approaches. As Table 1.1 indicates, the use of local vision is a challenge compared with global vision.

1.3 Research goal

The goal for this research is the development of local vision-based behaviors for the robotic soccer domain.

The scope of this research is delimited by the following restrictions:

1. The behaviors developed in this work are oriented to control a robotic soccer player with forward role.

2. A camera is the only sensor used by the robot.

3. The robot is equipped with a fixed gripper manually adapted to handle the ball.

4. The enclosed environment where the robot operates is a simplification of a soccer field.

5. There are not opponents for the robot. The robot and the ball are the only entities moving on the field.

6. The image processing system and the control system designed for the robot are executed by an external computer via wireless communication.

7. The camera mounted over the robot and the frame grabber installed in the external computer are connected through a cable.

8. Laboratory light conditions are controlled in a closed room by the use of artificial light.

To reach our goal we used a hierarchical control system [10] with a memory module, a local vision system, machine learning techniques and simple geometric calculations.

The control system was broke down into a set of behaviors which include: *finding ball, approaching ball, finding goal, approaching goal, shooting* and *avoiding*. For the *avoiding* behavior, we used the machine learning algorithm C4.5 [26] to learn whether a collision must be avoided or not. The rest of the behaviors were programmed by hand using simple geometric calculations.

The local vision system incorporates color space transformation from RGB to HUV, a pixel classification technique based on logical AND operations [13], region segmentation and object filtering. In addition, we test a pixel classification technique based on the Artificial Life paradigm.

1.4 Overview of the thesis

The thesis is organized in five chapters as follows:

The first chapter presents a brief introduction to robotic soccer, the difference between global and local vision, the objective followed by this research and the thesis organization.

The second chapter reviews related work in robotics, mobile robotics, robot control and robotic soccer.

The third chapter describes the perception and control system used by the robotic soccer player. A description of the robot used in this research and the environment which it operates. We describe the object detection system designed for the robot and a pixel classification method based on the Artificial Life paradigm. We also describe the architecture designed for the robot, this architecture is based on a hierarchical control system with a memory module.

The fourth chapter summarizes the results obtained during the research. A comparison between three pixel classification implementations is presented. We describe the method used to learn the *avoiding* behavior using the machine learning algorithm called C4.5 rule induction. We compare five machine learning algorithms used to learn the collision avoidance task. In order to test the global performance obtained by the robot, an experimental set was randomly designed. We present an analysis carried out over the experiments.

The fifth chapter discusses our conclusions derived of the research process, as well as possible perspectives of this work.

Chapter 2

Background

In this chapter, we present an introduction to robotics and mobile robotics. The most representative robot approaches to generate intelligent behavior are described: deliberative, reactive and hybrid. The RoboCup tournament is also described, including a brief RoboCup leagues description. Finally, we review related work on issues such as vision and control for RoboCup.

2.1 Robotics

The term *robot* was introduced by Karel Capek in his 1923 play Rossum's Universal Robots. The word robot is derived from the Czech word *robota*, meaning *labor*, and *robotnik*, meaning *slave*. The term robot has been applied to a great variety of mechanical devices, such as teleoperators, underwater vehicles, autonomous land rovers, industrial manipulators, and so on. In general, the term robot can be applied to any device that operates with some degree of autonomy, and that is usually controlled by a computer.

We present a number of definitions commonly used for describing a *robot*:

Definition 1: According to the Robotics Industry Association (*R.I.A.*) "a robot is a re-programmable, multi-functional manipulator designed to move materials, parts, tools, or specialized devices through variable programmed motions for the performance of a variety of tasks".

Definition 2: The Cambridge Advanced Learner Dictionary [1] defines robot as "a machine used to perform jobs automatically, which is controlled by a computer".

Definition 3: For Robin R. Murphy [23] "an intelligent robot is a mechanical creature which can function autonomously".

Definition 4: Rodney A. Brooks proposed this definition [12] "a situated creature or robot is one that is embedded in the world, and which does not deal with abstract descriptions, but through its sensors with the here and now of the world, which directly influences the behavior of the creature".

Definition 5: Ronald C. Arkin proposed [3] "an intelligent robot is a machine able to extract information from its environment and use knowledge about its world to move safely in a meaningful and purposive manner".

The definitions proposed by R.I.A. and the Cambridge dictionary are appropriate for industrial robots that are frequently programmed by hand to perform repetitive tasks.

The definition used by Murphy is focused on robots that operate without human intervention or autonomously, but in this definition the robot lacks of perception of the environment where it operates.

The definitions proposed by Brooks and Arkin are more appropriate for a robot that has to play soccer, because for them a robot should be able to sense the environment where it operates in order to extract the relevant information to accomplish its goals, then a sensor set should be considered in the equipment of the robot. Also, this relevant information can be transformed by the robot into useful knowledge to accomplish its goals. The robot is equipped with actuators then, it has the possibility to move safely on its environment.

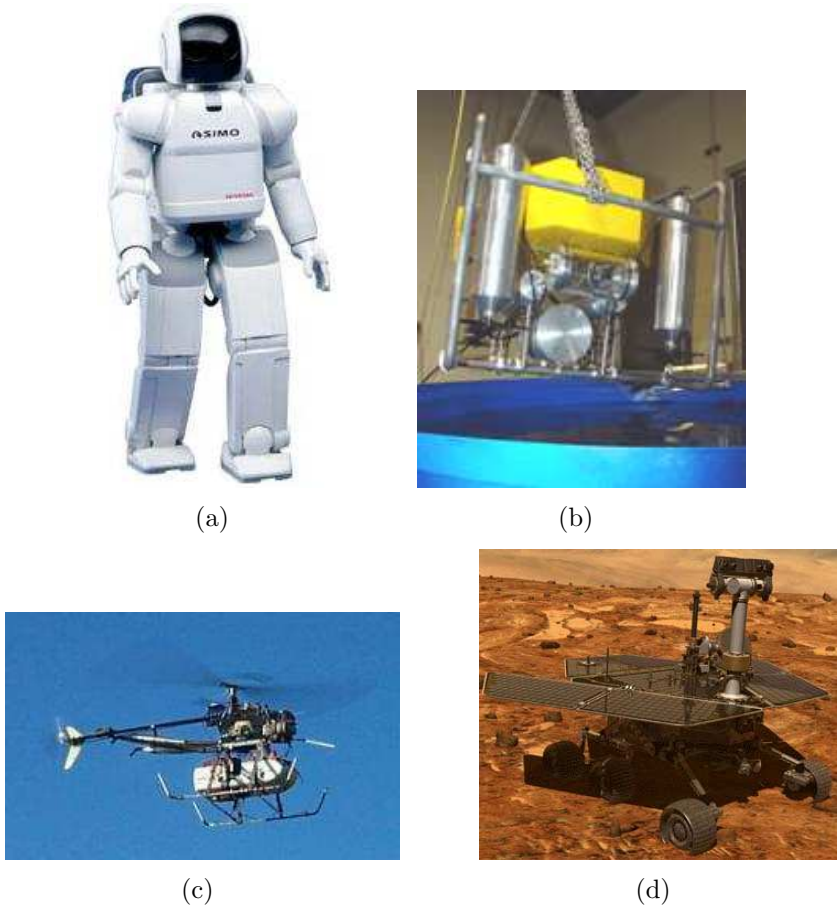
2.2 Mobile robotics

A mobile robot is a robot with the function to move from place to place autonomously. This robot is a combination of physical (hardware) and computational (software) components. The main characteristic of mobile robots is the locomotion process. A mobile robot can be considered as a collection of subsystems for:

- **Locomotion:** This subsystem moves the robot within its environment from place

to place.

- **Sensing:** This subsystem measures properties of itself and its environment.
- **Taking decisions:** This subsystem converts the measurements of the sensing subsystem into actions.
- **Communication:** This subsystem communicates the robot with an outside operator or with internal processes.



(a) (b)

(c) (d)

Figure 2.1: Mobile robots categories. (a) Terrestrial: Asimo (Advanced step in innovative mobility) is a human-like robot manufactured by Honda©from 1986 to 2000, with the intention of benefitting people in their daily life. (b) Aquatic: The robot for aquatic development and research©(RADAR) is a control system for an underwater remotely operated vehicle constructed by LSSU, in 2002-03. (c) Airborne: The CMU autonomous helicopter©was used to build 3D terrain models directly from its test flight data, in 1998. (d) Space: Mars exploration rovers created by NASA's Jet Propulsion Laboratory.

There are four broad categories of mobile robots, concerning the application domain. This categories are:

- **Terrestrial:** Terrestrial robots are those that move on the ground. They are designed to take advantage of a solid support surface and gravity. Although the most common terrestrial mobile robots are wheeled, also there are robots that can walk, climb, roll, use tracks, or slither in order to move. Terrestrial robots are also known as ground-contact robots. Figure 2.1(a) shows a picture of a terrestrial robot.

- **Aquatic:** Aquatic robots operate in water, either at the surface or underwater. Most existing aquatic vehicles use either water jets or propellers to provide locomotion. Aquatic robotics is a potentially important application domain because not only is most of the Earth's surface covered with water, but much of the ocean is not readily accessible to humans. Figure 2.1(b) shows a picture of an aquatic robot.

- **Airborne:** Airborne robot vehicles often mimic existing aircraft or birds. Robotic helicopters, fixed-wing aircraft, robotically controlled parachutes, and dirigibles have been developed. Flying robots share many issues with aquatic robots, including the need for energy even to remain stationary. Figure 2.1(c) shows a picture of an airborne robot.

- **Space:** Space robots are designed to operate in the micro-gravity of outer space and are typically envisioned for space station maintenance. Various locomotive devices enable robots in these environments to move according to the conditions of their environment. The two main classes of robot are those that move by climbing (over a larger vehicle) and those that are independently propelled (known as free flyers). Figure 2.1(d) shows a picture of a spatial robot.

2.3 Robot control

We reviewed different control architectures to generate intelligent behavior in mobile robots. A control architecture or robot architecture is the structure of organization of various modules: knowledge, internal representations, perception, actuation, processing and communication, that permit to reach the robot's objectives in interaction with its environment. For a control architecture, perception is fundamental to obtain the necessary information from the environment to reach the robot's objectives. The actuation allows the robot to interact with its environment via object manipulation o changing its position in the environment. Using the information obtained by the perception module, the robot can generate useful knowledge instead of unusable raw data obtained directly from its sensors. This knowledge can be represented internally and shared to other modules. Processing serves to convert the available information into actions. Finally, communication serves to establish communication channels with processes that include a human operator processes or other robots. The purpose of a control architecture is to permit the robot to replace the human operator in a task, playing soccer in our case.

There are three categories of robot architectures: deliberative, reactive and hybrid architecture. Similar classifications can be found in [3, 23]. We present a representative sample, as well as the advantages and disadvantages for each architecture.

The ideal control architecture does not exist [3, 23], only different designs or approximations to the solution of a kind of problem: an architecture for a robotic soccer player with forward role or for a goalkeeper, an architecture for a robot that has to push boxes, an architecture for a robot that recognizes the strategy taken by the opposing team.

2.3.1 Deliberative architecture

This architecture needs an explicit world representation to make decisions using this representation and following the rules of a planner. This architecture relies on the principle that complex problems can be subdivided into subproblems that are simpler and easy to solve. In this architecture, the best action to execute is always decided by the planner. For that, this planner has all the necessary information in the world

model created by the perceptive module. Deliberative architecture is also known as the sense-plan-act cycle (S-P-A), SMPA (sense-model-plan-act cycle) and the "top-down" paradigm. Figure 2.2 shows a diagram of the horizontal decomposition.

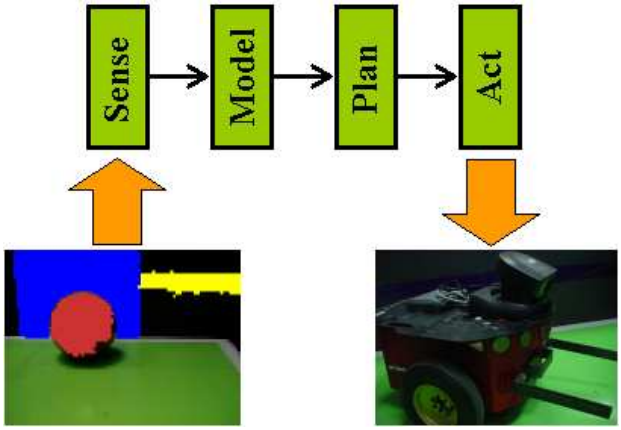


Figure 2.2: Horizontal Decomposition.

Functional decomposition (Modules):

- **Sensing:** This module perceives the world where the robot operates, it's the unique source of world information for the robot. This module is connected directly to the sensors.
- **Modelling:** This module takes the information collected in the sense module and it constructs a world model. A world model should be a symbolic representation of the environment, relevant information perceived by the robot or information about the robot's current situation.
- **Planning:** This module uses the world model constructed by the modelling module and it constructs an action plan for the robot. A plan is the sequence of steps necessary to accomplish a task.
- **Action:** This module uses the plan constructed in the planning module and it moves the robot according to that plan.

A representative example of the deliberative approach is the robot Shakey.

Deliberative example: architecture for the robot Shakey

Shakey the robot is considered the first mobile robot controlled by a computer using the deliberative paradigm. Figure 2.3 shows a picture of Shakey.

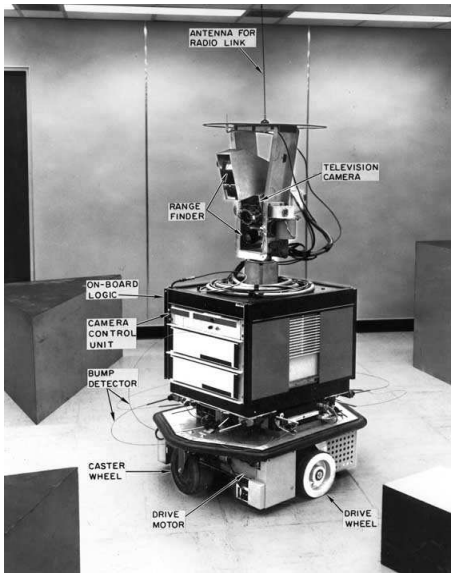


Figure 2.3: The robot Shakey.

This robot was constructed and programmed at Stanford Research Institute using symbolic artificial intelligence techniques. The robot was able to navigate in a room with obstacles and to push a block from point to point.

The architecture of Shakey was divided into 2 layers, the lower layer and the upper layer. The lower layer was constituted by some specialized functions, called action units. Each action unit had its own world model and planner, Shakey never stopped its execution until it finished successfully its task or until it detected a functional anomaly. The upper layer had the task of deliberating in a higher abstraction level. This layer performed logical deductions expressed in propositional logic over a world representation. The planner selected initially was the Question-Answering system (QA-3 Problem Solver) [18], based on methods for theorem proving. The planner called STRIPS [16] was incorporated later; this planner is a variation of the General Problem Solver (GPS) [24] based on the means-ends analysis [24]. STRIPS (STanford Research Institute Problem Solver) based its decisions over the selected operators in a table of differences in the following way: if the final state is not reachable with the use of one operator then

it selects the operator that reduces the difference between the next state predicted and the final objective. STRIPS worked recursively, chaining operators until it reached the goal. The logical effects of an operator are stored in list form, the post-condition list includes facts that stop being true (delete list) and new true facts (add list). Each operator required the satisfaction of a pre-conditions set, if all the preconditions were true then the operator was applied.

STRIPS implementations require the designer to set up a world model representation, a differences evaluator and a differences table with operators, preconditions, add and delete lists.

STRIPS has two main problems: the closed world assumption [25] and the frame problem [28]. The closed world assumption says that the world model contains everything the robot needs to know: there can be no surprises. The problem of representing a real-world situation in a way that was computationally tractable became known as the frame problem.

Figure 2.4 shows the architecture used by Shakey the robot.

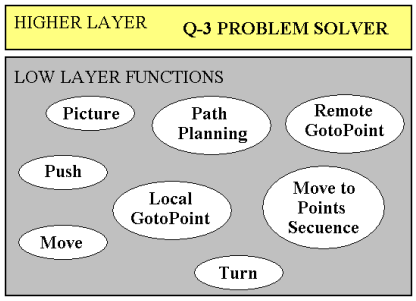


Figure 2.4: The robot Shakey architecture.

Deliberative architecture: advantages and disadvantages

- With the use of this architecture, the robot can have a predictive capacity. This predictive capacity is accomplished reasoning over the world model to predict an event before it occurs.

- The deliberative approach is a general method to solve a variety of tasks. The deliberative approach can use the same planner for every task by simply modifying

the knowledge base. This is independent of the domain of the problem and it's independent of the environment in which the robot operates.

- Reasoning produces correct results, but it is an expensive computational process.
- The main disadvantage in this architecture is the inability to react rapidly, this inability is generated by the reasoning process.
- This architecture assumes that the environment over which the robot operates is static, and all changes undergone by the environment are caused by the robot.
- Deliberative architecture requires a world model to perform a plan. This model must be updated quickly to prevent the planner from working with obsolete stored knowledge about the world.
- The functional decomposition in the deliberative architecture permits the designer to test the system until all the components are programmed.

2.3.2 Reactive architecture

Reactive control is based on the direct coupling of sensors to actuators. Its ideas are taken from ethology and biology. The reactive control paradigm is based on animal models of intelligence. In this paradigm, all actions are accomplished through modules called behaviors that are a direct mapping of sensory inputs to a pattern of motor actions that are used to achieve a task. It is common to associate a sensor output for each behavior that can process that output in a different way.

From a mathematical point of view, behaviors are a transfer function where a sensory input is transformed into a motor command. The use of a symbolic representation of the world and the use of a planner are not necessary in this paradigm.

The reactive paradigm suggests that intelligence is layered in a vertical decomposition, where the layers are connected to sensors and actuators independently. Different layers in the architecture take care of different behaviors. In the vertical decomposition the low level behaviors are used for survival, and the most advanced behaviors are oriented to solve high level goals.

The goals of a robot in reactive systems have no explicit representation, but these goals are encoded by individual behaviors. The overall behavior of the robot emerges from the interaction of behaviors with the environment.

This architecture is also known as the sense-act organization (S-A). Reactive architecture does not specify how the behaviors are coordinated or controlled. In fact, one behavior does not know what other behaviors are doing or perceiving. Figure 2.5 shows a diagram of the vertical decomposition.

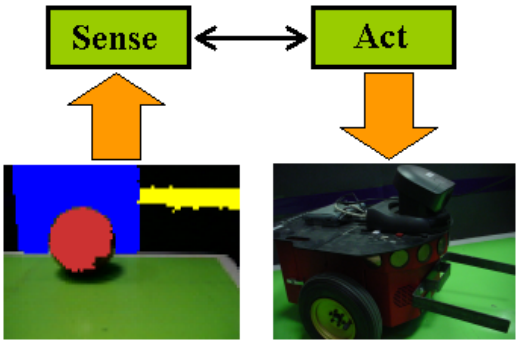


Figure 2.5: Vertical Decomposition.

Sensing in this paradigm is local to each behavior or behavior-specific. Each behavior has access to one or more sensors independently of the other behaviors. Behaviors can construct their own internal representations of the environment, but this representation is not a global world model such as in the deliberative paradigm.

A representative example of the reactive approach is the subsumption architecture [10].

Reactive example: subsumption architecture

The subsumption architecture [10] was developed by Rodney A. Brooks in the mid-1980s at the Massachusetts Institute of Technology.

The architecture decomposes the control system in behaviors grouped into layers of competence or hierarchies. The term behavior in this architecture is a network of sensing-acting modules which accomplish a task.

A finite-state machine (FSM)is an abstract computational element which is composed of a collection of states. Given a particular input, a FSM may change to a

different state or stay in the same state.

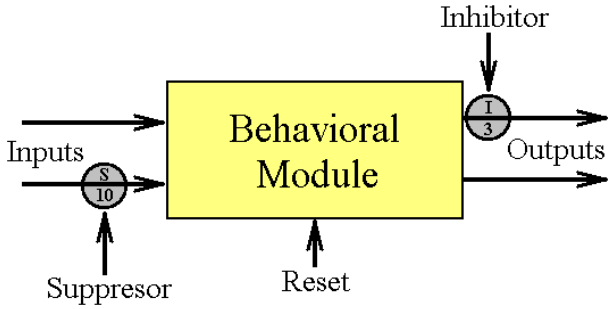


Figure 2.6: Augmented Finite State Machine used in the original subsumption architecture. An AFSM is an automata with timer; timers set changes after a previously established period of time has elapsed.

The modules are augmented finite state machines (AFSM), they are augmented because has registers, timers, and others enhancements to permit interface with other modules. Figure 2.6 shows the original AFSM used in the subsumption architecture proposed by Brooks.

The subsumption architecture introduced the concept of levels of competence [10]: "an informal specification of a desired class of behaviors for a robot over all environments it will encounter".

The subsumption architecture consists of a set of independent behaviors that directly map perception to action. The behaviors are organized in a static hierarchy, in which higher levels of competence have priority over lower levels. The control is not centralized, instead, the behaviors are executed in parallel.

Four interesting aspects of the subsumption architecture in terms of control are:

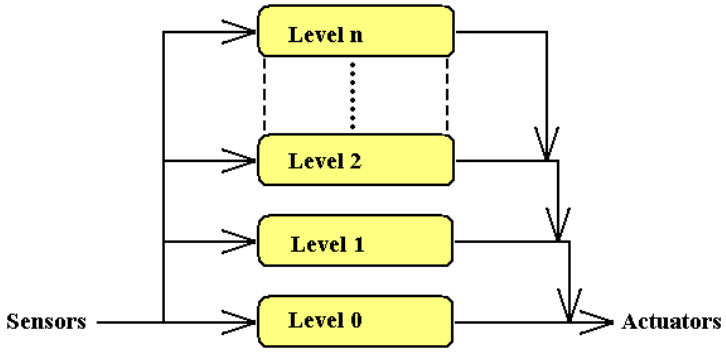


Figure 2.7: Competence levels of the subsumption architecture.

1. Behaviors are grouped into layers of competence. These layers reflect a hierarchy of intelligence, or competence. Lower layers are designated for survival tasks as avoiding collisions, while higher levels are designated for goal-directed actions such as map construction or shooting to goal. Each layer can be viewed as an abstract behavior for a particular task. Figure 2.7 shows the competence levels; the system can be partitioned at any level and the layers below form a complete control system.

2. Modules in a higher layer can override, or subsume, the output from behaviors in the next lower layer. This is the reason of the name for this architecture. Behavioral layers operate concurrently and independently, then a mechanism to avoid conflicts between layers is required. The solution to conflicts among layers in subsumption is the strategy of winner-take-all, where the absolute winner is always the higher layer in conflict.

3. The use of an internal state is avoided. In this case, an internal state means any type of representation of the state of the world or a world model. The architecture maps perception to action without any type of explicit or symbolic representation.

4. A task is accomplished by activating the correct layer at each moment. A layer is activated when a set of perceptual conditions are true and are deactivated otherwise.

In the subsumption architecture, the output from higher levels subsumes the output of lower levels. Subsumption can be done in two ways:

1. *Inhibition:* The output of the subsuming module is connected to the output of another module. If the output of the subsuming module is activated or has any value, the output of the subsumed module is blocked or deactivated.

2. *Suppression:* The output of the subsuming module is connected to the input of another module. If the output of the subsuming module is activated, it replaces the normal input of the subsumed module.

Much of the representation and style of the subsumption architecture is dogmatic. Ideas of this paradigm include:

- Complex behavior is not necessarily the product of a complex control system. Instead, the complex behavior emerges from the interaction of very simple behaviors. Intelligence is in the eye of the observer [11].
- The world is the best model for the robot [11], then a complex world representation is not necessary.
- Simplicity is a virtue, robots should be cheap and systems should be built incrementally.
- All onboard computation is important without complex computers and without high-bandwidth communication.

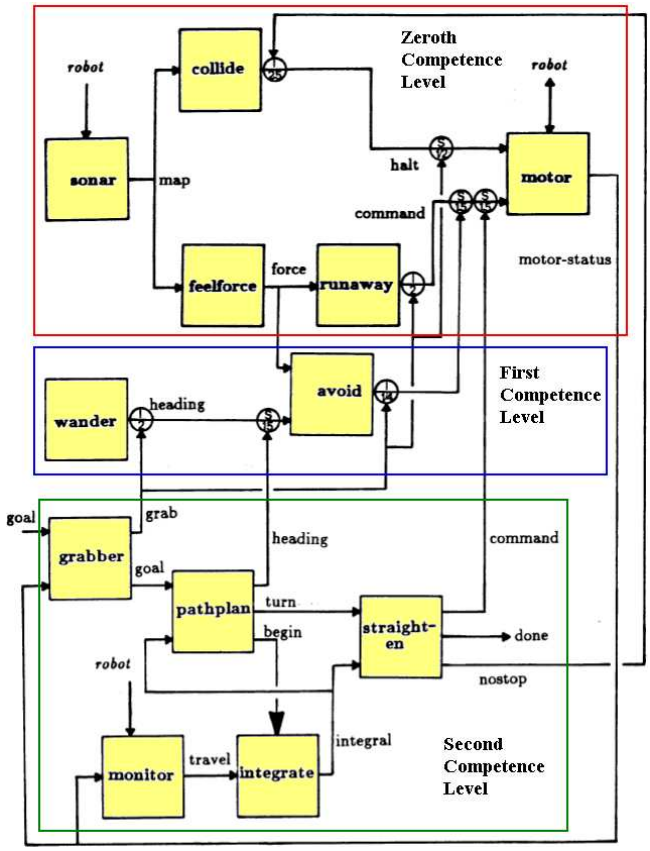


Figure 2.8: Subsumption architectural diagram example.

The example described by Brooks [12] is a robot that wanders the office areas of a laboratory, building maps of its surroundings. In this example, the behaviors are grouped in three competence levels. The lowest level layer of control or zeroth level has the task of avoiding collisions. The zeroth competence level is formed by the behaviors *motor, sonar, collide, feelforce* and *runaway*. The first level layer of control, when combined with the zeroth level gives the ability to the robot to wander around the laboratory without hitting obstacles. The first competence level is formed by the behaviors *wander* and *avoid*. Finally, the second level layer of control has the goal-oriented task, this is an exploratory mode where the robot navigates towards interesting places using visual observations. The second competence level is formed by the behaviors *grabber, monitor, pathplan, integrate* and *straighten*.

Figure 2.8 shows an architectural diagram with three competence levels used in the example described by Brooks.

Reactive architecture: advantages and disadvantages

- The main advantage of the reactive paradigm is the ability to react rapidly. In some applications, it is better to react rapidly than to execute the optimal action generated by a planner.

- The behaviors can be executed concurrently or sequentially. In this system the robot's overall behavior emerges from the interaction of the behaviors in the architecture.

- The modularity in this architecture permits the designer to use the existing behaviors as building blocks; complex behaviors may be assembled from primitive behaviors. These systems present good software engineering principles because of the "programming by behavior" approach. In this architecture, it is not necessary to construct the complete system to test it, the behaviors are tested incrementally and independently. The set of behaviors for the robot must be identified at design time.

- Reactive robotic systems have no memory, these systems are limited to react in a

stimulus-response style.

2.3.3 Hybrid architecture

It is a good idea to combine the reactive and deliberative approaches in order to overcome their disadvantages in a complementary way. This combination between reactive and deliberative subsystems gives as result a hybrid robotic control system

The main idea of the hybrid paradigm is based on two assumptions: First, planning covers a long time horizon and requires of global knowledge. Second, planning needs to be separated from real-time execution because it can slow down reaction time. Deliberation works with symbols, whereas reaction works with sensors and actuators.

Some researchers recommend that if a robot has been designed to operate in an unstructured environment, the designer should use the reactive paradigm. However, if the task was to be performed in a knowledge-rich environment and the knowledge is easy to model, then the deliberative paradigm is preferable.

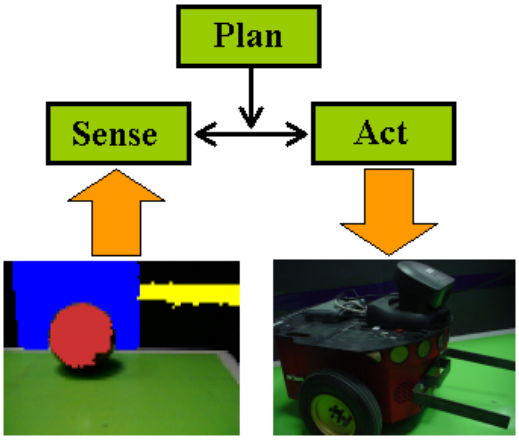


Figure 2.9: Hybrid architecture.

The organization of a hybrid system can be described as PLAN, then SENSE-ACT or P,S-A model. Figure 2.9 shows a general diagram of the hybrid architecture.

A representative example of the hybrid approach is the autonomous robot architecture (AuRA).

Hybrid example: Autonomous Robot Architecture (AuRA)

Autonomous Robot Architecture (AuRA) is based on schema theory [2] proposed by Arbib in 1992. AuRA is a hybrid architecture that incorporates deliberative and reactive (schema-based) control systems. The deliberative portion is a hierarchical planner based on traditional artificial intelligence techniques and the reactive portion is based on schema theory.

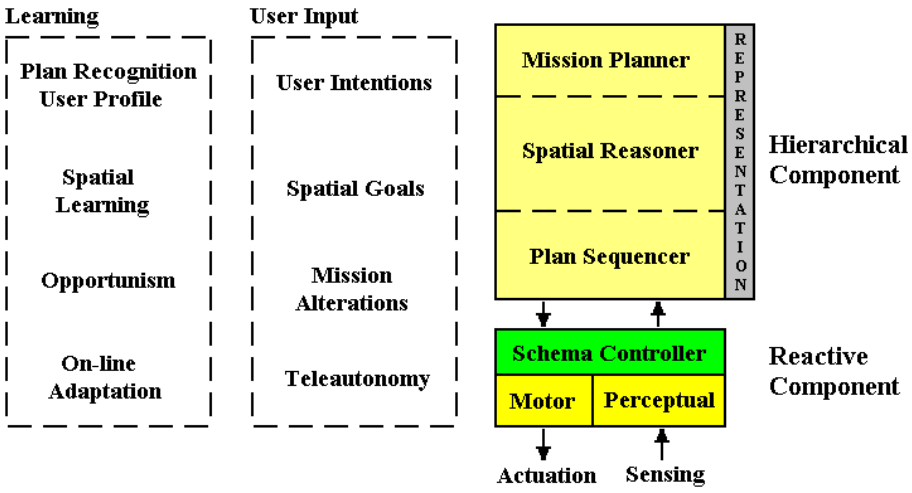


Figure 2.10: AuRA architectural diagram [3].

Figure 2.10 shows the AuRA architecture. The hierarchical system is composed by a mission planner, a spatial reasoner and a plan sequencer. The reactive system is a schema controller composed by the motor and perceptual subsystems.

In the same manner than in purely deliberative systems, the highest level of AuRA is the mission planner concerned with establishing high-level goals for the robot and the operational restrictions. The spatial reasoner or navigator, uses cartographic knowledge stored in long-term memory to construct a sequence of navigational paths that the robot must execute to complete its current mission assigned by the mission planner. The spatial reasoner encapsulates the map construction process and reading functions for navigating; it can be loaded with an *a priori* map. The plan sequencer or the pilot takes a subtask, it extracts relevant information for the subtask to generate motor behaviors for execution. The plan sequencer is the only connection with the reactive system, it gives the reactive system a list of behaviors to execute in order to accomplish its task.

The schema controller is responsible for controlling and monitoring the behavioral

processes at run time. Each motor behavior is associated with a perceptual schema that is able to provide the stimulus required for that particular behavior. Each behavior generates a response vector based on the potential field method [19] using bi-dimensional vectors; the overall behavior emerges by vector addition. The schemas operate asynchronously, transmitting their results to a process (move-robot) that sums and normalizes these inputs and transmits them to the low-level system for execution.

When the reactive system is activated, the deliberative system is deactivated until a failure is detected. The typical failure is denoted by lack of progress, a failure can be no motion detection or time-out. If a failure is detected, then the deliberative system is reactivated until the problem is solved.

AuRA has some strengths: it is modular, flexible and general. It is modular because the components of the architecture can be replaced with others. It is flexible because adaptation and learning methods can be introduced. It is generalizable to a wide range of problems that a purely reactive or deliberative architecture can not solve. The main strength of Aura is the advantageous combination of both hierarchical and reactive subsystems in one control system.

Hybrid architecture: advantages and disadvantages

- The main advantage of the hybrid architecture is the combination of the advantages of both approaches: reactive and deliberative. The goal is to design an architecture for a robot that is able to create a plan and execute it reacting rapidly.

- The robotics community has seen the hybrid architecture as the best general architectural solution for several reasons: First, we can use asynchronous processing techniques like multi-tasking and threads to allow deliberative functions to execute independently of reactive behaviors. Second, we can use modular software that allows subsystems in hybrid architectures to be mixed and matched for specific applications.

- The overall emergent behavior is a sequence of behaviors organized by the deliberative layer, and not the result of multiple reactive behaviors executing concurrently.

The overall emergent behavior in hybrid architectures is more predictable than in a purely reactive architecture.

- There is not a rule of thumb to create a hybrid architecture, although most hybrid systems provide a clear division between the functional and behavioral modes of control. There are no standard names for the levels of control in hybrid systems, although upper levels are often referred as strategic control or task control and lower levels are referred as execution control or reactive control. Designers are free to design a hybrid architecture, they are not limited for the rules imposed by each paradigm.

- Perception in hybrid architectures is more complex than in purely reactive or deliberative architectures. In the behaviors, perception has the same role than in the reactive paradigm: local and behavior-specific. Planning and deliberation require global world models to work. Planning functions have access to the world model. The construction of the world model is independent of the behaviors' perception.

- Reactive and deliberative subsystems share sensor readings in the hybrid approach. Moreover, the perception created by the behaviors can be used by the model-making processes.

- Hybrid architectures are task-dependent, the dependence consists in assigning a reactive portion and a deliberative portion to the hybrid architecture for a specific task. An architecture which had good performance for a task could not work well for another one, it depends on the given portions in the architecture.

2.3.4 Discussion

Robotic soccer has a highly dynamic environment where it is better to react rapidly than in the best way, for example, for avoidance tasks.

A robot having predictive capabilities in robotic soccer can anticipate some events that a pure reactive robot can not. For example, a robot can anticipate a collision some seconds before it occurs and can take corrective actions. A pure reactive robot knows

the goal's position until it perceives the goal, whereas a robot that has the position of the goal in its world model can use this knowledge to plan corrective actions.

In a robotic soccer player it is not possible to have a complete world model all the time, because unexpected elements such as team members or opponents can appear in the environment. However, a robotic soccer player that spends most of its time constructing the best plan to execute, can lose the reactivity necessary to execute simple actions such as avoiding other robots or to finding the ball.

Robot soccer is a domain where reactivity for a robot must be considered due to continuous and unexpected changes in the environment; however, a simplified world model or a deliberative subsystem integrated in reactive robot could help it to predict events before they occur. Given that deliberative and reactive architectures are complementary in order to overcome its disadvantages, the combination of both approaches in a hybrid control system is necessary for a robot that has to play soccer.

2.4 Robot soccer

The World RoboCup Challenge [21], or RoboCup, is an international competition for robotic soccer teams. It has been held every year since 1997. The purpose of RoboCup is to provide a standard problem for AI and Robotics research [20].

The main goal of RoboCup according to Asada and Kitano [5] is: "By the mid 21st century, a team of autonomous humanoids robots shall beat the human World Cup champion team under the official regulations of FIFA".

The RoboCup is an event of special interest for researchers, because it requires the integration of a broad range of technologies in order to integrate and control a robotic soccer team. Some research areas that RoboCup covers are [5]:

- Agent and Multi-agent architectures in general.
- Combining reactive approaches and modeling/planning approaches.
- Real-time recognition, planning and reasoning.
- Sensor-fusion.

- Behavior learning for complex tasks.
- Strategy acquisition.

In 1993, Alan Mackworth first proposed to build robots for playing soccer [22]. Mackworth proposed: "let us consider a world in which all assumptions of traditional AI were violated, suppose we want to build a robot to play soccer".

2.4.1 League description

We present an overview of the robotic soccer leagues that form the RoboCup. There are currently 5 robotic soccer leagues in the RoboCup Soccer Competition, these leagues are: *simulation league, small-size league, middle-size league, Sony four-legged robot league* and *humanoid league*.

1. **Simulation league:**

   In RoboCup Simulation League, the teams are composed of 11 autonomous agent players that use the RoboCup Soccer Server Simulator.

Figure 2.11: RoboCup 2003 simulator snapshot.

   In this league there are no physical robots, but spectators can watch the action on a large screen. Each simulated soccer player may have its own game strategy and own characteristics and every simulated team consists of a collection of programs.

Many computers are connected with a main server in order for this competition to take place. The games last 10 minutes and the half-time 5 minutes. A snapshot of the RoboCup 3D Simulator is shown in Figure 2.11.

2. **Small-size league:**

   Small-size robot soccer, or F180 as it is otherwise known, focuses on the problem of intelligent multi-agent cooperation and control in a highly dynamic environment with a hybrid centralized/distributed system. Building a successful team requires clever design, implementation and integration of many hardware and software sub-components into a robustly functioning whole making small-size robot soccer a very interesting and challenging domain for research and education.

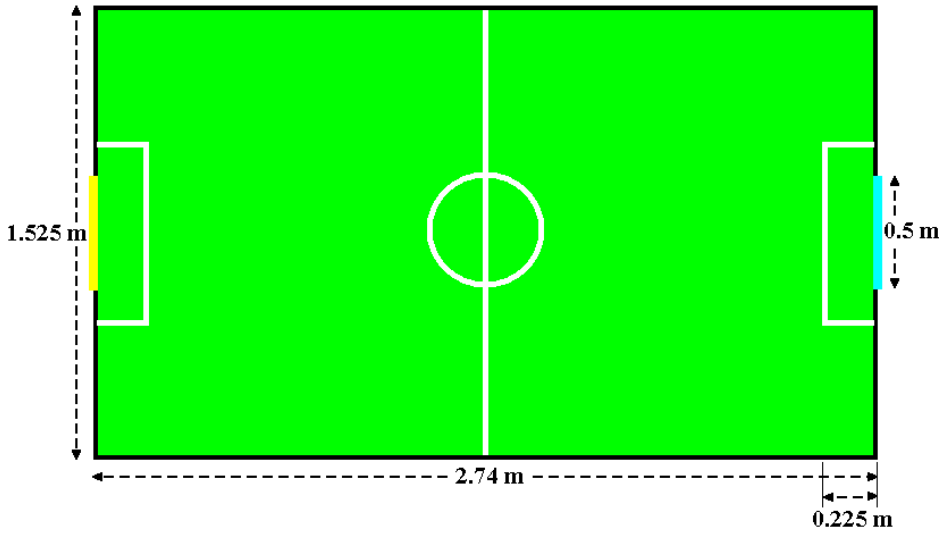


Figure 2.12: Small-size field dimensions.

   A Small-size team is composed of no more than 5 robots. The field has the size of a ping pong table (1.525 m $\times$ 2.74 m). The corners are designed in such a way that it prevents the ball or robots from getting stuck. The goals are 0.5 m wide, goals are painted blue and yellow and the field is painted green. A standard orange golf ball is used. Global vision (see, section 1.2). The ping pong table dimensions are shown in Figure 2.12.

3. **Middle-size league:**

   The major difference to the small-size league is that no global vision of the field

is allowed, hence the robots carry their own sensors, including vision. The robots are fully autonomous, i.e., all their sensors, actuators, power supply and (in most cases) computational power are on-board, and no external intervention by humans is allowed, except to insert or remove robots in/from the field. External computational power is allowed, even though most teams do not use it. Wireless communications among the team robots and/or with the external computer are also allowed. The areas of interest in the middle-size league are: Self-localization, multi-agent coordination, learning in dynamic environments, cooperative autonomous behaviors in multi-robot systems, distributed sensor fusion, vision-based perception, autonomous navigation. A Middle-size team is composed of no more than

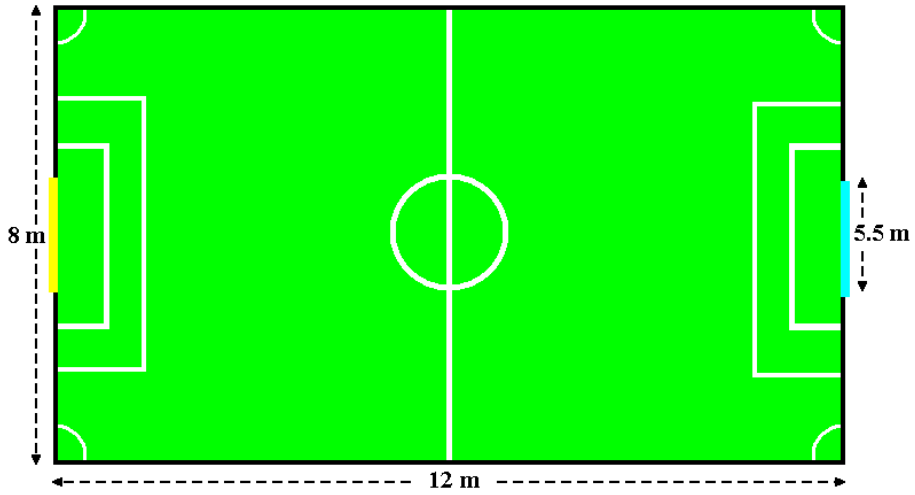


Figure 2.13: Middle-size field dimensions.

11 robots. The field has the size (8 m $\times$ 12 m). The goals are 5.5 m wide, goals are painted blue and yellow and the field is painted green. The official tournament ball used in matches is any FIFA standard size 5 football. The color for the ball is orange. The base color of the robots must be black. The paint must be matte in order to reduce reflexivity. Robots must have markings in order to be recognized by other robots and to be distinguished by the referee. Global vision is prohibited, the robots need to use local vision (see, section 1.2). Communication between robots is allowed. Any kind of perception system is allowed, if it is integrated on the robot. Robots may have special devices for ball handling, if they never prevent the ball from rolling freely. The match lasts two equal periods of 45

minutes, unless otherwise mutually agreed between the referee and the two participating teams. The half-time interval must not exceed 10 minutes. Robots may dribble the ball either by making repeated or continuous physical contact with the ball. A robot may kick the ball by making a single, short physical contact with the ball. The field dimensions are shown in Figure 2.13.

4. **Sony four-legged robot league:**

   The four-legged robot league is similar as the middle-size league in relation to challenges. The main difference is the robot platform, AIBO robots. AIBO robots have legs instead wheels.

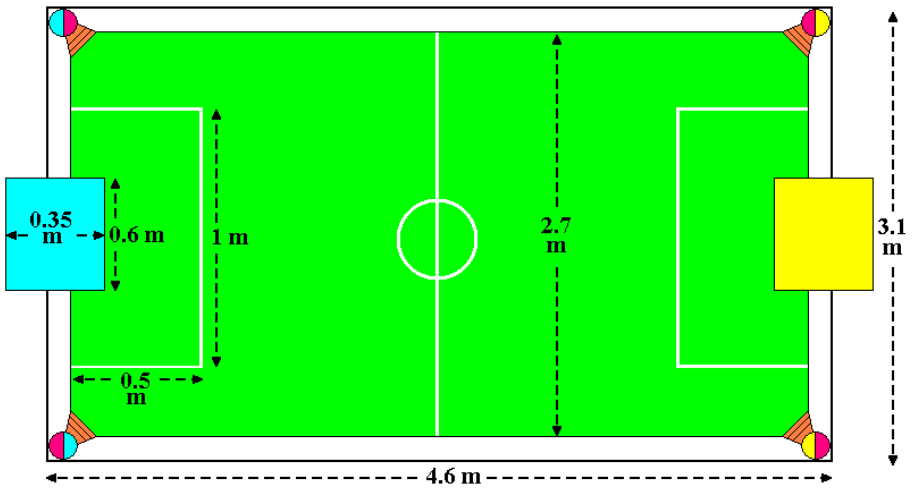


Figure 2.14: Sony Four-Legged field dimensions.

A Four-legged team is composed of no more than four AIBO ERS robots including the goal keeper. The field has the size of 4.6 m × 3.1 m. The goals are 0.6 m wide, goals are painted blue and yellow and the field is painted green. There are 4 cylindrical beacons in each corner with a different color combination, the robots can use it to determine their positions in the field. The walls that surround the field and each triangular corner has 45°slanted. The purpose of this slope is to return the ball to the field when the ball is pushed against the border or the corners. The field dimensions are shown in Figure 2.14. The color of the ball is orange.

The robots need to use team markers to be recognized by teammates. The colors

Figure 2.15: AIBO is the first entertainment robot created by Sony©.

of the markers are red and blue. An AIBO robot is shown in Figure 2.15. Global vision is prohibited, the robots use local vision.

5. **Humanoid league:**

Figure 2.16: Robo-Erectus (RE) is a low-cost autonomous humanoid robot being developed at ARICC (the Advanced Robotics and Intelligent Control Centre) in the School of Electrical and Electronic Engineering, Singapore Polytechnic. The current challenge for the Robo-Erectus project is to develop a team of low-cost soccer-playing humanoid robots that are capable of walking, turning, crouching, and kicking a football past a goalkeeper into the goal.

This league has some challenges which include walking, balancing, passing, obstacle avoidance, passing and human control with high degree of freedom systems. A humanoid robot should be able to walk using two legs, no wheels are allowed. A humanoid robot must consist of two legs, two arms, one body, and one head. There are 3 classes of humanoid robots: H-40, H-80 and H-120. The ball for humanoids of class H-40 and H-80 is the one used for four-legged robots. The ball for humanoids of class H-120 is the one used for middle-size robots. The field size is 7.2 m by 10.8 m. A humanoid robot is shown in Figure 2.16.

2.4.2 Related work in robot soccer

We survey a number of works in the field of vision and control for robotic soccer domain.

Vision

Template matching was used by Cheng and Zelinsky [15] in the vision system for their autonomous soccer robots. In template matching, objects can be identified by comparing stored object templates against the perceived image. Template matching can fail if the light intensity varies significantly over areas where the template is applied.

The cognachrome vision system©, manufactured by Newton Research Labs is a commercial hardware-based vision system used by several robot soccer teams [31]. Since it is hardware-based, it is faster than software running on a general-purpose processor. Its disadvantages are its high cost and the fact that it only recognizes three different color classes.

A number of past RoboCup teams have used alternative color spaces such as HSB or HSV for color discrimination proposed by Asada [4], since these separate color from brightness reducing sensitivity to light variations.

Several RoboCup soccer teams have adopted the use of omnidirectional vision generated by the use of a convex mirror [7]. This type of vision has the advantage of providing a panoramic view of the field, sacrificing image resolution. Moreover, the profiles of the mirrors are designed for a specific task.

The fast and cheap color image segmentation for interactive robots employs region segmentation by color classes [13]. This system has the advantage of being able to classify more than 32 colors using only two logical AND operations and it uses alternative color spaces.

For our vision system, we used the pixel classification technique proposed by Bruce [13] and a variant of the color spaces proposed by Asada [4] (see Section 3.2).

Control

Takahashi et al. [30] used multi-layered reinforcement learning, decomposing a large state space at the bottom level into several subspaces and merging those subspaces at

the higher level. Each module has its own goal state, and it learns to reach the goal maximizing the sum of discounted reward received over time.

Steinbauer et al. [29] used an abstract layer within their control architecture to provide the integration of domain knowledge such as rules, long term planning and strategic decisions. The source of action planning was a knowledge base that contained explicit domain knowledge used by a planning module to find a sequence of actions that achieves a given goal.

The RMIT RoboCup team [14] used a symbolic model of the world. The robot can use it to reason and make decisions.

Bonarini et al. [8] developed a behavior management system for fuzzy behavior coordination. Goal-specific strategies are reached by means of conflict resolution among multiple objectives. Behaviors can obtain control over the robot according to fuzzy activation conditions and motivations that reflect the robot's goals and situation.

Bredenfeld et al. [9] used the "dual dynamics" model of behavior control. This is a mathematical model of a control system based on behaviors. The robot's behaviors are specified through differential equations.

Gómez et al. [17] used an architecture called dynamic schema hierarchies. In this architecture, the control and the perception are distributed on a schema collection structured in a hierarchy. Perceptual schemas produce information that can be read by motor schemas to generate their outputs.

We used a behavior-based control system or subsumption architecture [10] with a memory module in order to control our robotic soccer player (see Section 3.3).

Chapter 3

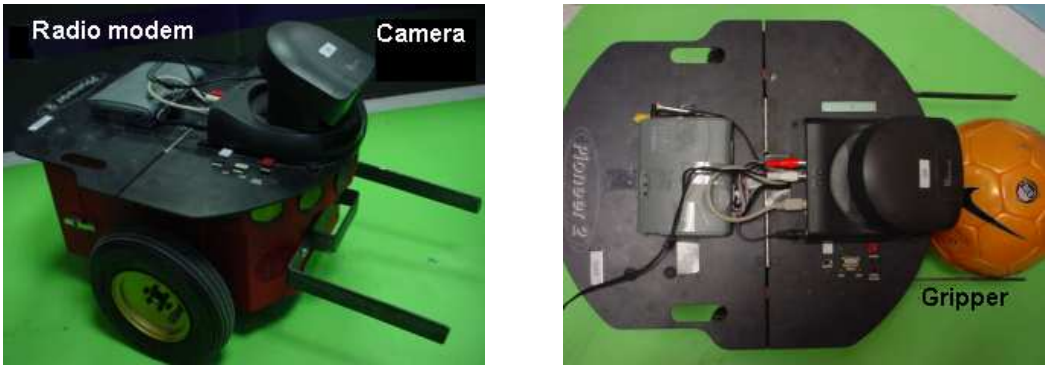
The system

In this chapter, we describe the vision system and control architecture used by our robotic soccer player. In vision, we present the object detection system steps and pixel classification method based on Artificial Life. In control architecture, we add a memory module for the subsumption architecture [10]. Finally, we present a functional description of the behaviors.

3.1 Hardware and settings

The robot used in this research is a Pioneer 2-DX mobile robot made by Activ-Media©, equipped with a Pioneer PTZ camera, a manually-adapted fixed gripper and a radio modem. The dimensions of the robot are 44 cm long, 38 cm wide and 34 cm tall, including the video-camera. The robot is remotely controlled by a AMD Athlon 1900 computer with 512 MB of RAM. Figure 3.1 shows two pictures of our robotic soccer player.

The environment for the robot is an enclosed playing field with a size of 180 cm in length and 120 cm in width. There was only one goal, painted cyan, centered in one end of the field with a size of 60 cm wide and 50 cm tall. The walls were marked with an auxiliary purple line whose height is 20 cm from the floor. Figure 3.2 shows a picture of the playing field.



(a) (b)

Figure 3.1: The robotic soccer player. A lateral view (a). A superior view (b).

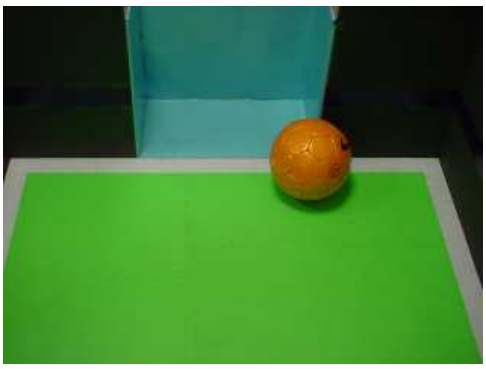


Figure 3.2: The soccer playing field.

3.2 Vision

A robust, fast and fault tolerant vision system is fundamental for the robot, because it is the only source of information about the state of the environment. Since all objects of interest in the environment are colored, we believe that vision is the most appropriate sensor for a robot that has to play soccer. We present below the object detection system used by the robot and a strategy for pixel classification based on the Artificial Life paradigm.

3.2.1 Object detection

The vision system processes images captured by the robot's camera and reports the locations of various objects of interest relative to the robot's current location. The objects of interest are the orange ball, the cyan goal and the auxiliary purple line on

the field's wall. The steps of our object detection method are:

1. *Image capture:* Images are captured in RGB in a 160×120 resolution.

2. *Image resizing:* The images are resized to 80×60 pixels.

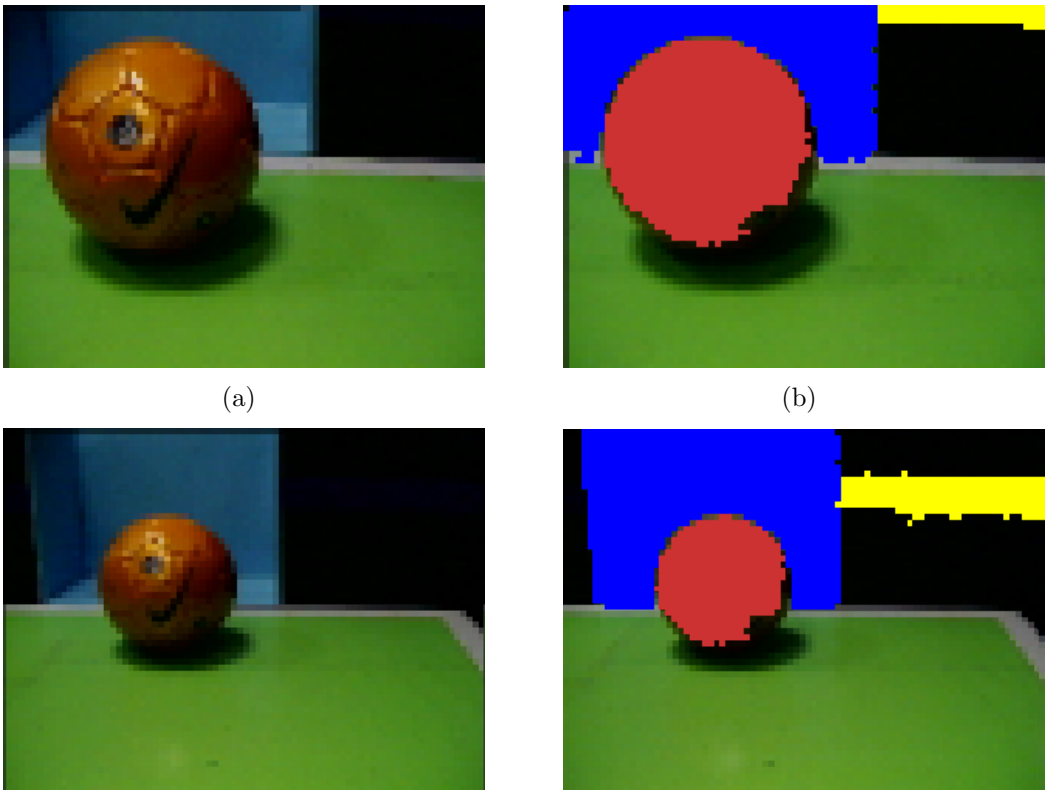
3. *Color space transformation:* The RGB images are transformed into the HUV color space, for reducing sensitivity to light variations.

4. *Pixel classification:* Each pixel is classified by predetermined color thresholds in RGB and HUV color spaces. There are 3 color classes: the colors of the ball, the goal, and the auxiliary line. The pixel classification is based on [13], in order to use only two logical AND operations for each color space.

5. *Region segmentation:* Pixels of each color class are grouped together into connected regions.

6. *Object filtering:* False positives are filtered out via region size.

We are using the smallest resolution (80×60 pixels) for the images where an object of interest is distinguished by its color, size and form. 80×60 resolution was enough for object detection. The color space transformation is a necessary step because we can classify pixels with a minimum and maximum threshold and the light sensitivity is reduced. The region segmentation step is used to consider groups of pixels that have more probability to be part of an object of interest instead of isolated pixels that can be noise. The object filtering step discards small segments of grouped pixels because they can be noise in the image. If the number of pixels that form a region is bigger than a preestablished threshold then the region is considered as an object of interest. The final result of the color classification is a new image indicating the color class membership for each pixel. Figure 3.3 on the left column shows two images captured by the frame grabber and on the right column shows the respective robot's perception.



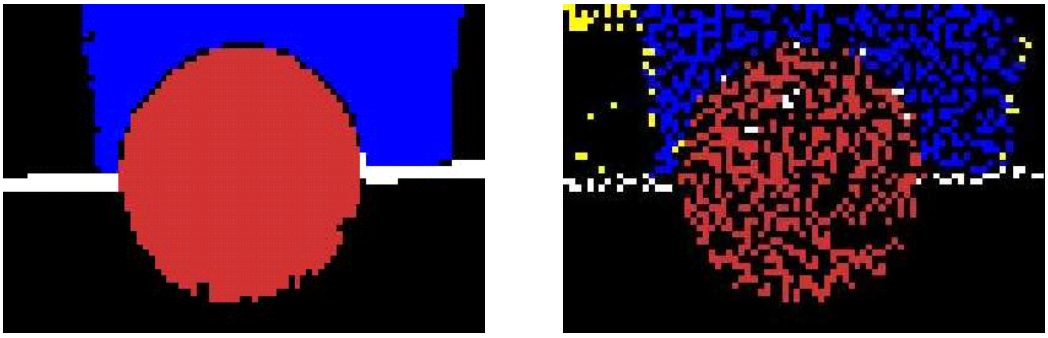
(a) (b)

(c) (d)

Figure 3.3: Images captured by the camera (a) and (c). The robot's perceptions (b) and (d). In the robot perception images, the ball is painted red, the goal is painted blue and the auxiliary line is painted yellow.

3.2.2 Artificial Life for pixel classification

In order to reduce the time invested in pixel classification, the most expensive step in object detection, we tested an Artificial Life-based method. Ideas of distributed computing were taken from Reynolds's boids [27], where a group of agents moves as a flock of birds or a school of fish.



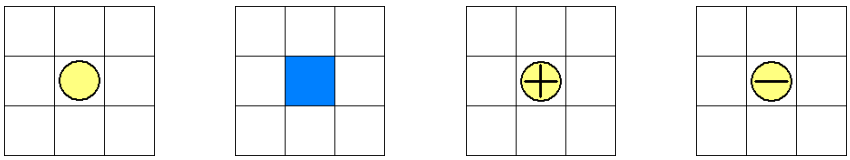
(a) (b)

Figure 3.4: Pixel classification snapshots with both approaches. Bruce-based (a) and Artificial Life-based (b).

For this strategy, we used 2500 agents, each having an internal state to indicate whether it is over a pixel of an object of interest or not. Agents were able to detect 3 color classes: the colors of the ball, the goal and the auxiliary line in the walls. Agents were serialized by an agent manager which assigned movement turns and prevented collisions between agents. The agents can move in their world, which is the image perceived by the camera. Only one agent can be situated over each pixel, thus agents are distributed over the image. Agents can sense the color intensity values in the image in order to perform pixel classification. The locomotion of an agent consists of moving pixel by pixel via its actuators. Figure 3.4 shows a snapshot of the pixel classification method based on Artificial Life.

Agents' movements

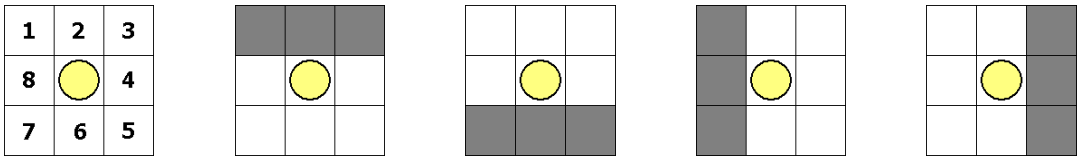
A candidate pixel is a pixel which is part of an object of interest. An agent is attracted by a candidate pixel then its movements are guided by these pixels. A candidate pixel is painted gray. An agent that is situated in a candidate pixel is represented by the "+"



(a) (b) (c) (d)

Figure 3.5: Agent descriptions. (a) An agent which is calculating its next movement is represented as a circle. (b) A candidate pixel. (c) An "+" agent situated in a candidate pixel. (d) An "−" agent.

sign. An agent that is not situated in a candidate pixel is represented by the "−" sign. Figure 3.5 shows an agent, a candidate pixel, an "+" agent and an "−" agent.



(a) (b) (c) (d) (e)

Figure 3.6: (a) Eight possible pixels for moving. (b)-(e) Pixel groups used to calculate the directions to north, south, east and west respectively.

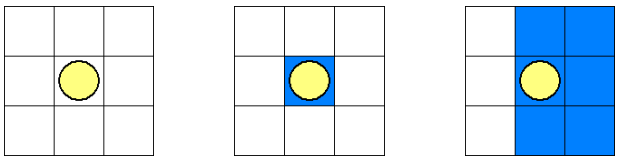
Figure 3.6(a) shows the eight possible pixels where an agent can perform a movement. Gray blocks in Figures 3.6(b), 3.6(c), 3.6(d) and 3.6(e) represent the pixel groups used calculate the movement's force of an agent towards north, south, east and west respectively. If an "+" agent is situated in a gray cell then 1 must be added. If an "−" agent is situated in a gray cell then 1 must be subtracted.

To decide the final direction between north and south directions, the greater value is selected. The same principle is applied to calculate the final direction between east and west directions. If final direction is 0 there is not any movement.

If final direction selected between north and south is greater than zero and final direction selected between east and west is greater than zero then both directions are combined to generate a diagonal movements which include northeast, northwest, southeast and southwest.

When calculating its next movement, an agent can face two situations. First case, there are not agents around itself. Second case, there is at least one agent around itself.

First case: There are not agents around an agent which is calculating its next



(a) (b) (c)

Figure 3.7: Three possible movements of an agent where there are not agents around it. Pixels are painted white and candidate pixels are painted blue.

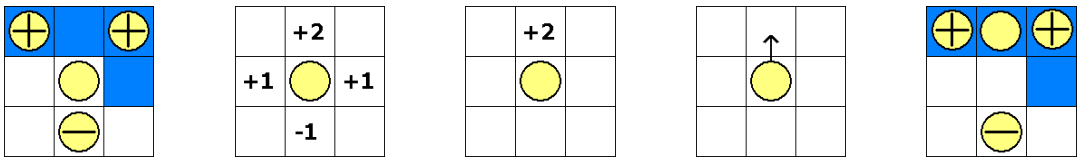
movement; this agent can perform three possible movements:

1. If an agent is not located on a candidate pixel and there are not candidate pixels around itself then the agent moves randomly. Figure 3.7(a) shows this case.

2. If an agent is located on a candidate pixel and there are not candidate pixels around itself then the agent remains in the same pixel. Figure 3.7(b) shows this case.

3. If an agent is located on a candidate pixel and there are candidate pixels around itself then the agent moves randomly to a candidate pixel. Figure 3.7(c) shows this case.

Second case: There is at least one agent around an agent which is calculating its next movement; this agent will move closer to agents whose internal state indicates that they are situated on a candidate pixel and this agent will move away to agents whose internal state indicates that they are not situated on a candidate pixel. The agent can perform the following movements:

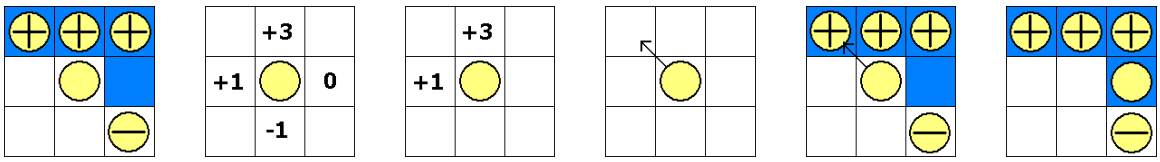


(a) (b) (c) (d) (e)

Figure 3.8: Simple gather.

1. *Simple gather,* where agents come together in group. Figure 3.8 describes the calculations performed by an agent which is performing the *simple gather* movement.

Figure 3.8(a) shows the initial state. Figure 3.8(b) shows the movement's forces for north, south, east and west directions. Figure 3.8(c) indicates that north movement must be performed and that east and west movements are cancelled each other out. Figure 3.8(d) indicates the candidate pixel. In Figure 3.8(e) the agent performs its next movement towards the candidate pixel because that pixel is not occupied by other agent.

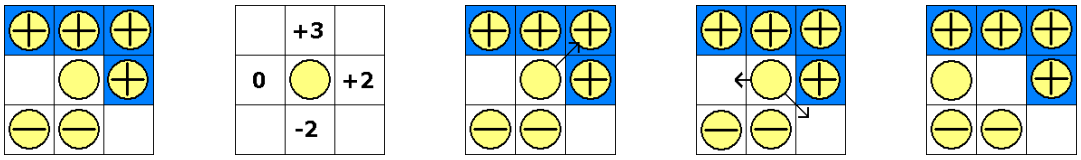


(a) (b) (c) (d) (e) (f)

Figure 3.9: Gather and avoid.

2. *Gather and avoid,* where agents try to come together in group, but the candidate pixel is occupied by another agent. Figure 3.9 describes the calculations performed by an agent which is performing the *gather and avoid* movement.

   Figure 3.9(a) shows the initial state. Figure 3.9(b) shows the movement's forces for north, south, east and west directions. Figure 3.9(c) indicates that north and east movements must be performed. Figure 3.9(d) indicates the candidate pixel. Figure 3.9(e) shows that the candidate pixel is occupied for an "+" agent. In Figure 3.9(f) the agent performs its next movement towards a candidate pixel randomly chosen.



(a) (b) (c) (d) (e)

Figure 3.10: Crowd around group.

3. *Crowd around group,* where agents try to come together in group, but each candidate pixel is occupied by another agent. Figure 3.10 describes the calculations performed by an agent which is performing the *crowd around group* movement.

Figure 3.10(a) shows the initial state. Figure 3.10(b) shows the movement's forces for north, south, east and west directions. Figure 3.10(c) shows that the candidate pixel is occupied for an "+" agent. Figure 3.10(d) shows two possible pixels to perform the agent's next movement. In Figure 3.10(e) the agent performs its next movement towards an empty pixel randomly chosen.

3.3 Control

Behaviors were designed and implemented using a subsumption architecture [10] because this architecture offers the necessary reactivity for dynamic environments. We incorporated a new element to this architecture, a memory module. This module acts as a short-term memory that enables the robot to remember past events that can be useful for future decisions. The memory module affects directly the behaviors programmed into the robot.

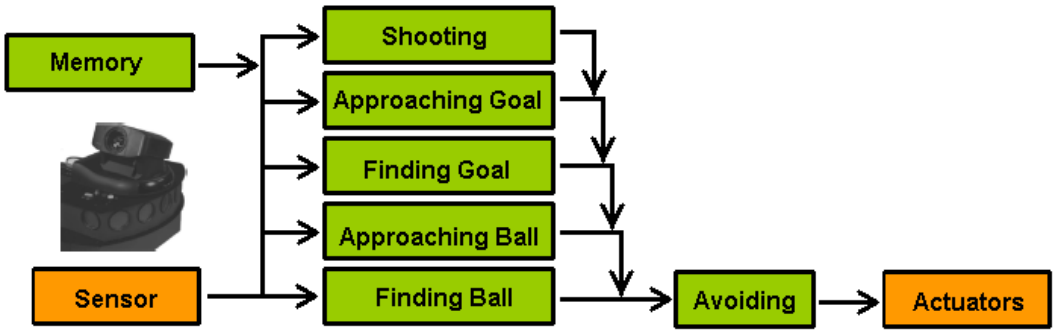


Figure 3.11: The architecture of the system.

The *avoiding* behavior is a horizontal behavior in the architecture that overwrites the output of the rest of the behaviors in our vertical subsumption architecture. The architecture was implemented using four threads in C++, one for the vertical behaviors module, one for the memory module, one for controlling the robot movements and one for the horizontal behavior to avoid collisions with the walls. In this architecture, each behavior has its own perceptual input, which is responsible of sensing the objects of interest. Each behavior writes its movement commands to shared memory to be executed. The architecture used for the robot's control system is shown in Figure 3.11.

3.3.1 Description of modules and behaviors

1. *Memory:* This is an essential module for the achievement of the robot's global behavior. Memory, like behaviors, has its own perceptual input to sense the ball and the goal. The function of this memory is to remember the last direction in which the ball or the goal were perceived with respect to the point of view of the robot. The memory module affects directly the other behaviors because it writes the directions of the ball and the goal on a shared memory used in the behaviors's execution. There are six possible directions that the memory has to remember: ball to the left, ball to the right, centered ball, goal to the left, goal to the right and centered goal.

2. *Finding ball:* The robot executes a turn around its rotational axis until the ball is perceived. The robot turns in the direction in which the ball was last perceived. If this information was not previously registered then the robot executes a random turn towards the left or right.

3. *Approaching ball:* The robot centers and approaches the ball until the ball is at an approximate distance of 1 cm.

4. *Finding goal:* The robot executes a turn around its rotational axis until the goal is perceived. The robot turns in the direction in which the goal was last perceived. If this information was not previously registered then the robot executes a random turn towards the left or right.

5. *Approaching goal:* The robot executes a turn in the direction of the center of the goal until the goal is centered with respect to the point of view of the robot.

6. *Shooting:* The robot makes an abrupt increase of its velocity to shot the ball towards the goal. There are two possible kind of shots, a short shot when the robot is close to the goal (a distance equal or less than 65 cm) and a long shot, when the robot is far from the goal (more than 65 cm).

7. *Avoiding:* The robot avoids crashing against the walls that surround the soccer field. Determining manually the necessary conditions in which the robot collides

with the wall is difficult because the wall can be perceived in many forms, therefore we used the machine learning algorithm C4.5 [26] to learn whether a collision must be avoided or not (for results, see section 4.2).

3.3.2 Global behavior

The global behavior of our robotic soccer player is described by the automaton in Figure 3.12.

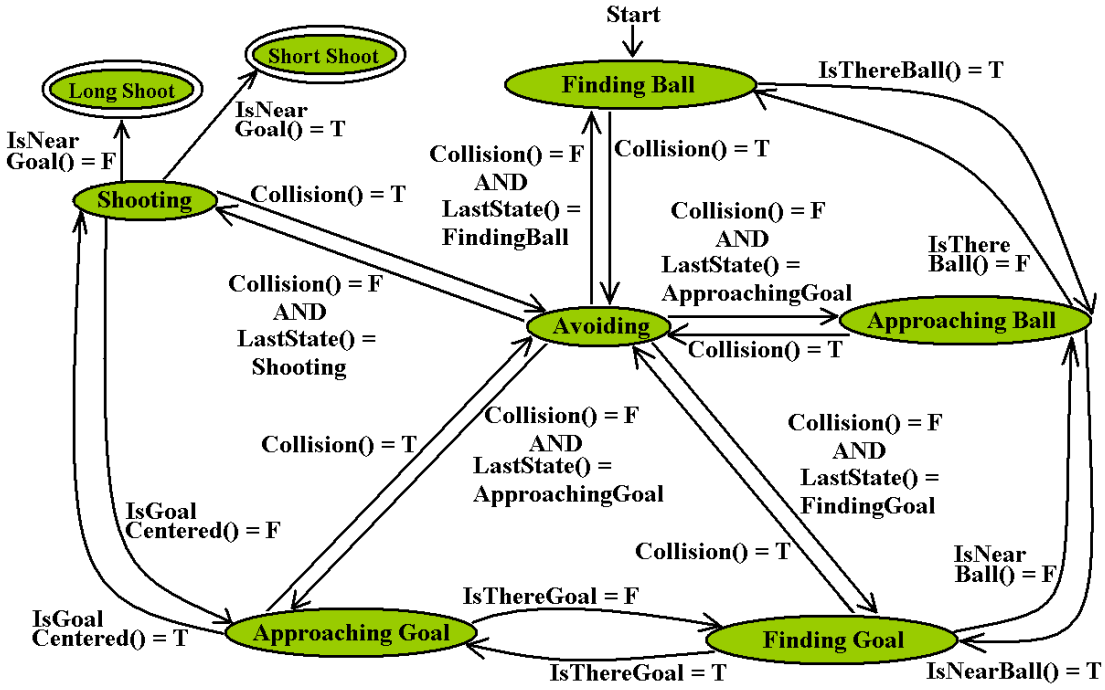


Figure 3.12: Automaton resuming the global behavior of the robot. IsThereBall(), IsNearBall(), IsThereGoal(), IsGoalCentered(), IsNearGoal() and Collision() are boolean functions to indicate: if the ball is perceived, if the ball is near to the robot, if the goal is perceived, if the goal is centered with respect to the point of view of the robot, if the goal is near (a distance equal or less than 65 cm) to the robot and if a collision with the walls is detected, respectively. The return values for the boolean functions are: T (true) or F (false). The function LastState() returns the last state visited in the automaton.

Chapter 4

Experimental results

In this chapter, we summarize the experimental results obtained by the robotic soccer player described in the previous chapter. We present the results obtained by three implementations of pixel classification methods, the results obtained for the learning behavior, *avoiding*. Finally, we performed a set of 20 random experiments in order to test the global performance of the robotic soccer player.

4.1 Pixel classification results

We present the results obtained by three implementations of pixel classification. The first implementation was performed based on linear color thresholds, the second implementation was based on the algorithm proposed by Bruce et al. for pixel classification [13], and finally, the third implementation was based on the Artificial Life paradigm.

Method	Images per second	Processing average time
Linear color thresholds	12 images	0.0874 sec.
Bruce-based method	18 images	0.0553 sec.
Artificial Life-based method	14 images	0.0707 sec.

Table 4.1: Pixel classification results.

Results of pixel classification are shown in Table 4.1. As this table indicates, the worst strategy for pixel classification task was based on linear color thresholds. The best strategy for this task was based on the algorithm proposed by Bruce et al. [13], this strategy was implemented as a step in the object detection system for the robotic soccer player. We expected a better performance from the pixel classification method based on

Artificial Life, because this method needs to examine only 2500 pixels, corresponding to the total number of agents, instead of the total number of pixels in the image (4800 pixels). However, in this strategy each of the agents spends time calculating its next movement, producing a general medium performance according to Table 4.1.

4.2 Avoiding behavior results

For the *avoiding* behavior, we collected a training set of 446 instances of collisions. There were 153 positive samples where there was a collision and 293 negative samples where there was no collision. The elements of the input vector were *roundness, compactness, convexity, orientation, contour length, mean length of runs, line index of lower right corner point, column index of lower right corner point, row of the largest inner circle* and *column of the largest inner circle* of the auxiliary line's region detected by the object detection system. The experiments were validated using 10-fold cross-validation.

Machine learning algorithm	% of correctly classified instances
Support Vector Machines	91.25 ± 0.0603 %
Artificial Neural Networks	90.40 ± 0.0849 %
C4.5	92.20 ± 0.0638 %
Naive Bayes	87.62 ± 0.0683 %
Conjuntive Rules	90.68 ± 0.0222 %

Table 4.2: Percentage of correctly classified instances by machine learning algorithm for the *avoiding* behavior.

We tested 5 machine learning algorithms for the classification task; the results obtained are summarized in Table 4.2. As this table shows, the C4.5 algorithm obtained the best percentage of correctly classified instances for the collision avoidance task. The rules generated by C4.5 algorithm were implemented in our *avoiding* behavior, these rules are shown in Figure 4.1.

```
J48 pruned tree
------------------

dsrRow21 <= 31
|   dPhil <= -0.195257
|   |   dsrColumn21 <= 42: 0 (2.0)
|   |   dsrColumn21 > 42
|   |   |   dPhi <= 2.920391: 1 (12.0)
|   |   |   dPhi > 2.920391: 0 (3.0/1.0)
|   dPhil > -0.195257
|   |   dContLength <= 208.870056
|   |   |   dMeanLength <= 12.299213
|   |   |   |   dicRow <= 10: 0 (23.0/1.0)
|   |   |   |   dicRow > 10: 1 (2.0)
|   |   |   dMeanLength > 12.299213: 0 (242.0)
|   |   dContLength > 208.870056
|   |   |   dConvexity <= 0.698842: 1 (4.0)
|   |   |   dConvexity > 0.698842
|   |   |   |   dCompactness <= 2.466859: 1 (4.0/1.0)
|   |   |   |   dCompactness > 2.466859: 0 (10.0)
dsrRow21 > 31
|   dsrRow21 <= 34
|   |   dRoundness <= 0.53691: 1 (43.0/3.0)
|   |   dRoundness > 0.53691
|   |   |   dContLength <= 224.225403
|   |   |   |   dicColumn <= 17: 0 (2.0)
|   |   |   |   dicColumn > 17: 1 (9.0/1.0)
|   |   |   dContLength > 224.225403: 0 (7.0)
|   dsrRow21 > 34: 1 (83.0/1.0)

Number of Leaves  :     14

Size of the tree :      27
```

Figure 4.1: Rules obtained for the avoiding behavior. Class 1 indicates collision and class 0 indicates no collision.

Figure 4.2 shows 15 images included in the positive training set and Figure 4.3 shows 15 images included in the negative training set.

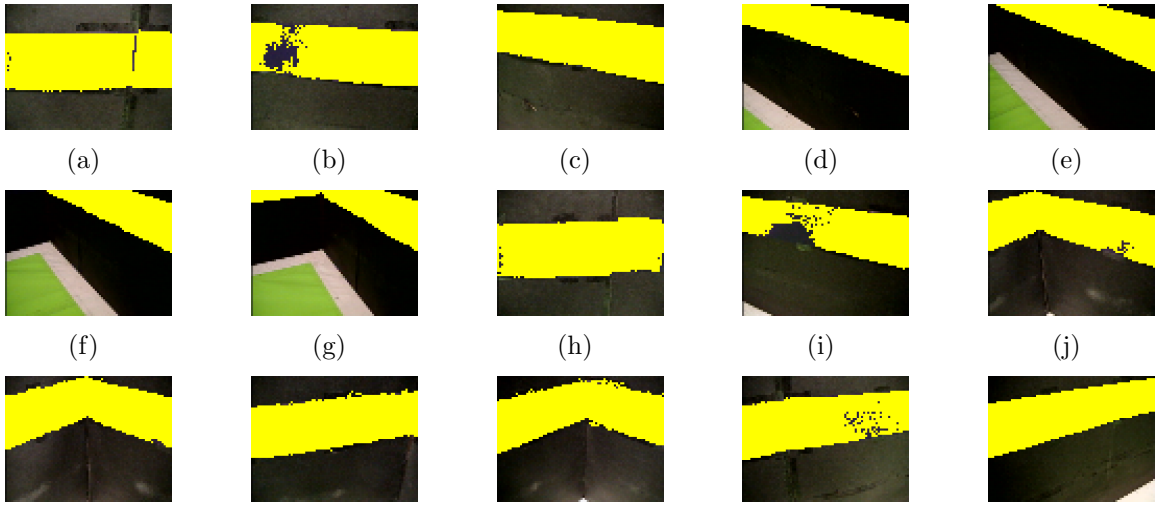


Figure 4.2: Positive training set used to learn the *avoiding* behavior.

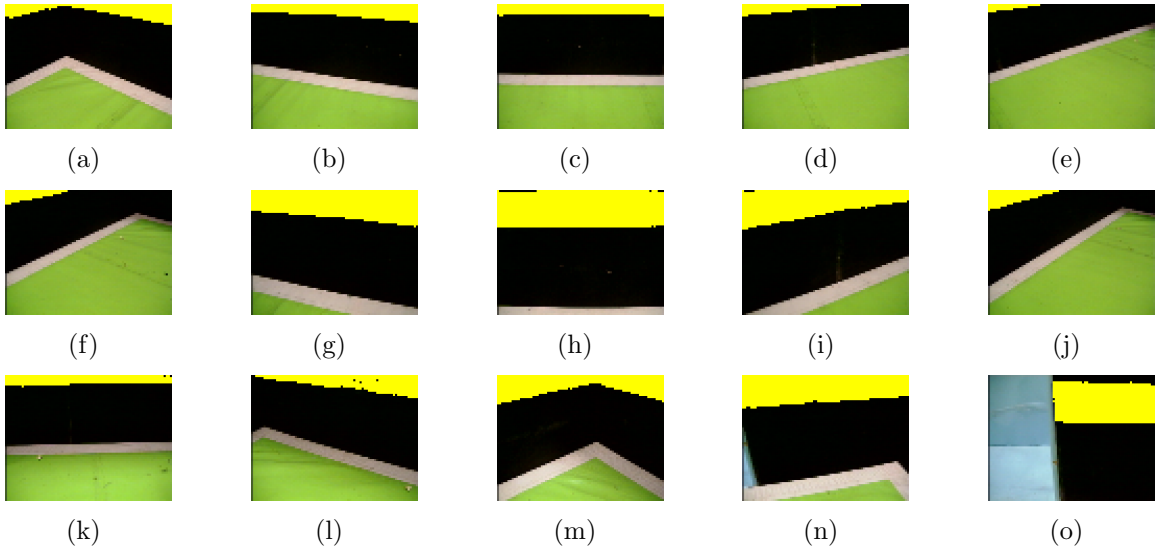
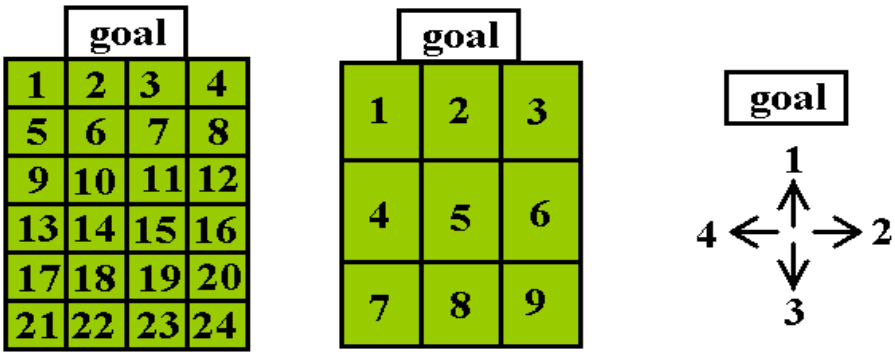


Figure 4.3: Negative training set used to learn the *avoiding* behavior.

4.3 Global performance

Our robotic soccer player has a forward role, thus its main task is to score goals in a minimum amount of time. In order to test the global performance of the robot, we designed a set of experiments. The experiments were performed on the soccer field shown in Figure 3.2. The robot position, robot orientation and ball position were selected 20 times randomly as follows:



(a) (b) (c)

Figure 4.4: Experimental configuration cells. Ball position (a). Robot position (b). Robot orientation (c).

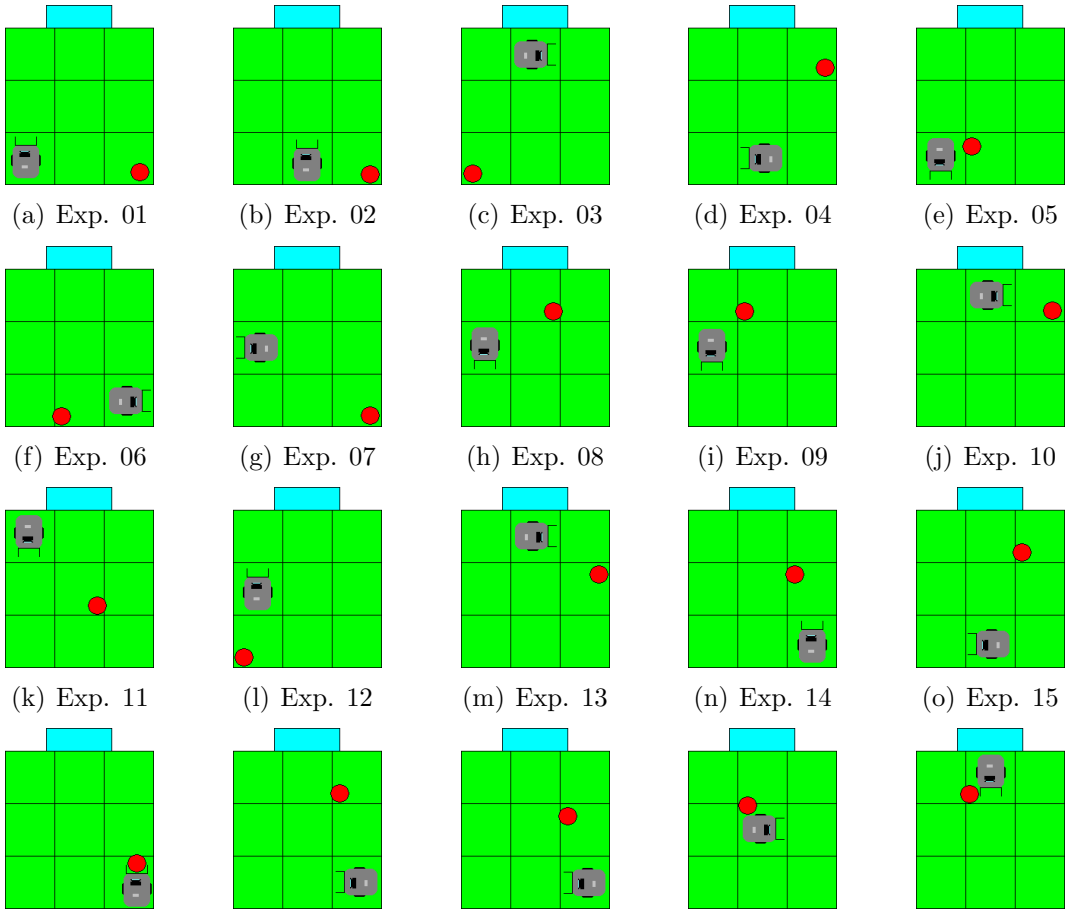
1. For selecting the ball position, the field was divided into 24 cells of equal size. Figure 4.4(a) shows the cells for the ball position.

2. For selecting the robot position, the field was divided into 9 cells of equal size. Figure 4.4(b) shows the cells for the robot position.

3. For selecting the robot orientation, there were 4 directions to the robot. The orientation where the goal is: 1) in front of the robot, 2) left to the robot, 3) back to the robot and 4) right to the robot. Figure 4.4(c) shows the possible orientations for the robot.

4. We selected a division of 9 cells of equal size to situate the robot position, because it is the number of robots which can stay on the field at the same time. We can situate 3×3 robots over the field. It is the same case for selecting the ball position. In the case of the robot orientation, we did not consider diagonal orientations of the robot with respect the goal.

An experiment's configuration can be represented as a triplet, of the form *(ball position, robot position, robot orientation)*. The configuration for the 20 experiments performed were: *(24,7,1)*, *(24,8,1)*, *(21,2,2)*, *(8,8,4)*, *(18,7,3)*, *(22,9,2)*, *(24,4,4)*, *(7,4,3)* , *(6,4,3)*, *(8,2,2)*, *(15,1,3)*, *(21,4,1)*, *(12,2,2)*, *(11,9,1)*, *(7,8,4)*, *(20,9,1)*, *(7,9,4)*, *(11,9,4)*, *(10,5,2)* and *(6,2,3)*. Figure 4.5 shows the experimental configuration diagrams representing these experiments.



(a) Exp. 01 (b) Exp. 02 (c) Exp. 03 (d) Exp. 04 (e) Exp. 05

(f) Exp. 06 (g) Exp. 07 (h) Exp. 08 (i) Exp. 09 (j) Exp. 10

(k) Exp. 11 (l) Exp. 12 (m) Exp. 13 (n) Exp. 14 (o) Exp. 15

(p) Exp. 16 (q) Exp. 17 (r) Exp. 18 (s) Exp. 19 (t) Exp. 20

Figure 4.5: Experimental configuration diagrams. The ball is in a corner in 5 experiments: (a), (b), (c), (g) and (l). Exp. means experiment number.

Experiment Number	Finding Ball	Approaching Ball	Finding Goal	Approaching Goal	Shooting	Duration
1	16	10	10	–	6	42
2	11	19	17	1	6	54
3	–	–	–	–	–	–
4	13	9	3	4	5	34
5	8	5	–	2	6	21
6	8	11	7	10	6	42
7	6	31	6	3	6	52
8	5	9	7	1	5	27
9	5	6	–	5	5	21
10	8	6	–	4	5	23
11	2	16	8	–	6	32
12	17	20	11	6	6	60
13	–	–	–	–	–	–
14	–	7	–	3	6	16
15	16	11	–	7	5	39
16	–	5	–	–	6	11
17	27	8	–	4	6	45
18	19	8	–	3	6	36
19	6	12	11	2	5	36
20	10	11	10	5	5	41
Totals	177	204	90	60	101	632 sec

Table 4.3: Time spent, in seconds, in each of the behaviors executed by the robot during 20 experiments. The symbol "-" indicates that a behavior in a given experiment was not executed. An experiment that contains only "-" symbols, is an unsuccessful experiment in which the robot got stuck against the walls

Table 4.3 summarizes the time spent in seconds by each behavior performed by the robot in the experiments. The total time spent for the robot in the experiments was 632 seconds.

The percentage of time used by behaviors in the experiments was in general 28% for *Finding Ball*, 32.27% for *Approaching Ball*, 14.24% for *Finding Goal*, 9.49% for *Approaching Goal* and 16% for *Shooting*. The average time required by the robot to score a goal is 35.11 seconds.

The performance of *avoiding* behavior was acceptable, the robot avoided 10 of 12 avoidance situations obtaining 83% success.

A useful functionality of the soccer player emerges from the interaction of 3 behaviors: *approaching ball*, *finding goal* and *avoiding*. This emergent behavior consists of

regaining the ball from the corner. In the experiments, the robot was able to *regain the ball from the corner* four out of five times obtaining 80% success. In the 20 experiments executed, the robot was able to score 18 goals obtaining 90% success.

Chapter 5

Conclusions and future work

5.1 Conclusions

This research focused on the development of local vision-based behaviors for a Pioneer 2-DX robotic soccer player. For achieving our goal we used a hierarchical control system with a memory module, a local vision system, machine learning techniques and simple geometric calculations.

The subsumption architecture used for the robot control gives the necessary reactivity to play soccer. Even though the robot displays a highly reactive behavior, the memory that we incorporated enables the robot to base its decisions on past events.

Building the *avoiding* behavior using the C4.5 algorithm to learn to avoid collisions with the walls, was much easier to learn than to program by hand. The performance of *avoiding* behavior in the experiments was acceptable, the robot avoided 10 out of 12 avoidance situations obtaining 83% success.

Although the strategy for pixel classification based on Artificial Life did not improve the performance, it seems to be a promising strategy to create a completely distributed control system for a robotic soccer player. The main limitation of this approach is the current computational processing power needed to support a large number of agents with complex behaviors.

Using our object detection method we can detect the ball, the goal and the auxiliary line, at a frame rate of 17 frames per second.

Experimental results obtained with our robotic soccer player indicate that the robot operates successfully showing a high-level intelligent behavior and scoring goals in 90%

of the trials.

5.2 Future work

In future work we will use other machine learning techniques, such as artificial neural networks or support vector machines, to help us develop behaviors such as *approaching ball*, because these techniques have been used to learn complex task such as vehicle driving and object recognition.

For the avoidance behavior, we need to collect more training samples where the robot is near the corner. We think that increasing the training set, the robot could avoid getting stuck in the corners when it is regaining the ball.

Our robotic soccer player increases currently its velocity to shoot the ball towards the goal. We consider to implement in the near future a kicking device to improve the shooting behavior.

Our incremental tendency for constructing a robotic soccer team is based on the design of a group of specialized robots in a particular task, for example: a robotic soccer player with forward role, a defence or a goalie. The next step to reach in our research is to consider multi-robot coordination by designing and implementing a top-level coordination mechanism to control several robots as an intelligent robotic soccer team.

The architecture used in this work could have several applications which include:

- *Cleaner robot:* The cleaner robot should find the garbage to clean instead to find the soccer ball. Also, the robot could place the picked garbage in a garbage dump instead to shoot the ball to the goal. For this application we could use our object detection system to detect several kinds of garbage and the garbage dump. We must incorporate a mechanism for garbage picking.

- *Rescue robot:* The rescue robot should find a person in danger inside a building instead to find the soccer ball. Also, the robot could approach that person instead to approach to the ball. For this application we could use our object detection

system to detect the walls, obstacles and persons. Finally, the robot could take that person to a safe place.

Publications

1. **A. Salim and O. Fuentes and A. Muñoz**. *Development of local perception-based behaviors for a robotic soccer player.* Proceedings of IX Iberoamerican Conference on Artificial Intelligence. To appear in Lecture Notes in Computer Science. 2004.

2. **A. Salim and O. Fuentes and A. Muñoz**. *Development of local vision-based behaviors for a robotic soccer player.* Proceedings of the Mexican International Conference in Computer Science. To appear. 2004.

3. **A. Salim and O. Fuentes and A. Muñoz**. *Development of local vision-based behaviors for a robotic soccer player.* Technical report CCC-04-005. National Institute of Astrophysics, Optics and Electronics. Computer Science Department. 2004.

Bibliography

[1] *The cambridge advanced learner dictionary*, Cambridge University Press, 2003.

[2] M. A. Arbib, *Schema theory*, The Encyclopedia of Artificial Intelligence (1992), 1427–1443.

[3] R. C. Arkin, *Behavior-based robotics*, The MIT Press, 1998.

[4] M. Asada and H. Kitano (eds.), *Robocup-98: Robot soccer world cup ii*, Lecture Notes in Computer Science, vol. 1604, Springer-Verlag, 1999.

[5] ———, *The robocup challenge*, Robotics and Autonomous Systems (1999), 3–12.

[6] M. Asada, P. Stone, H. Kitano, B. Werger, Y. Kuniyoshi, A. Drogoul, D. Duhaut, and M.Veloso, *The robocup physical agent challenge: Phase-i*, Applied Artificial Intelligence **12** (1998), no. 2-3.

[7] A. Bonarini, P. Aliverti, and M. Lucioni, *An omnidirectional vision sensor for fast tracking for mobile robots*, Transactions on Instrumentation and Measurement **49** (2000), 509–512.

[8] A. Bonarini, G. Invernizzi, T. Labella, and M. Matteucci, *An architecture to coordinate fuzzy behaviors to control an autonomous robot*, Fuzzy Sets and Systems **134** (2003), 101–115.

[9] A. Bredenfeld, T. Christaller, W. Göhring, H. Günter, H. Jaeger, H. Kobialka, P. Plöger, P. Schöll, A. Siegberg, A. Streit, C. Verbeek, and J. Wilberg, *Behavior engineering with "dual dynamics" models and design tools*, RoboCup-99 Team Descriptions Middle Robots League Team GMD Robots (1999), 134–143.

[10] R. A. Brooks, *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation (1986), no. RA-2.

[11] ———, *Intelligence without reason*, 12th International Joint Conference on Artificial Intelligence (1991), 569–595.

[12] ———, *Flesh and machines: how robots will change us*, Pantheon Books, 2000.

[13] J. Bruce, T. Balch, and M. Veloso, *Fast and inexpensive color image segmentation for interactive robots*, In Proc. of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (2000), 2061–2066.

[14] J. Brusey, A. Jennings, M. Makies, C. Keen, A. Kendall, L. Padgham, and D. Singh, *RMIT robocup team*, RoboCup-99 Team Descriptions Middle Robots League Team RMIT (1999), 181–188.

[15] G. Cheng and A. Zelinsky, *Real-time vision processing for a soccer playing mobile robot*, Lecture Notes in Computer Science **1395** (1997), 144–155.

[16] R. E. Fikes and N. J. Nilsson, *STRIPS: a new approach to the application of theorem proving to problem solving*, Artificial Intelligence (1971), 189–208.

[17] V. M. Gómez, J. M. Cañas, F. San Martín, and V. Mantellán, *Vision-based schemas for an autonomous robotic soccer player*, Proceedings of IV Workshop de Agentes Físicos WAF-2003 (2003), 109–120.

[18] C. Green, *Application of theorem proving to problem solving*, First International Joint Conference on Artificial Intelligence (1981), 202–222.

[19] O. Khatib, *Real-time obstacle avoidance for manipulators and mobile robots*, 1985 IEEE International Conference on Robotics and Automation (1990), 500–505.

[20] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, *Robocup: The robot world cup initiative*, IJCAI-95 Workshop on Entertainment and AI/Alife, 1995.

[21] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, *Robocup: A challenge problem for AI and robotics*, RoboCup-97: Robot Soccer World Cup I (1997), 1–19.

[22] A. Mackworth, *On seeing robots*, Computer Vision: System, Theory, and Applications (1993), 1–13.

[23] R. R. Murphy, *Introduction to AI robotics*, The MIT Press, 2000.

[24] A. Newell and H. Simon, *GPS: A program that simulates human thought*, Computers and Thought (1963), 279–293.

[25] D. L. Poole, A. K. Mackworth, and R. G. Goebel, *Computational intelligence: A logical approach*, Vancouver, B.C.:Dept. of Computer Science, University of British Columbia (1992), 383.

[26] J. R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann, 1993.

[27] C. W. Reynolds, *Flocks, herds, and schools: A distributed behavioral model*, Computer Graphics **21** (1987), 25–34.

[28] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, Prentice Hall, 1995.

[29] G. Steinbauer and M. Faschinger, *The mostly harmless robocup middle size team*, OGAI **22** (2003), 8–13.

[30] Y. Takahashi and M. Asada, *Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning*, IEEE/RSJ International Conference on Intelligent Robots and Systems **1** (2000), 395–402.

[31] B. B. Werger, P. Funes, M. Schneider-Fontán, R. Sargent, C. Witty, and T. Witty, *The spirit of Bolivia: Complex behavior through minimal control*, Lecture Notes in Computer Science **1395** (1997), 348–356.

List of Figures

List of Tables