Autonomous Soccer Robots

Wei-Min Shen, Jafar Adibi, Rogelio Adobbati, Bonghan Cho,
Ali Erdem, Hadi Moradi, Behnam Salemi, Sheila Tejada
Computer Science Department / Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292-6695
email: {shen,dreamteam}@isi.edu
URL: http://www.isi.edu/isd/dreamteam

Abstract. The Robocup 97 competition provides an excellent opportunity to demonstrate the techniques and methods of artificial intelligence, autonomous agents and computer vision. On a soccer field the core capabilities a player must have are to navigate the field, track the ball and other agents, recognize the difference between agents, collaborate with other agents, and hit the ball in the correct direction. USC's Dreamteam of robots can be described as a group of mobile autonomous agents collaborating in a rapidly changing environment. The key characteristic of this team is that each soccer robot is an autonomous agent, self-contained with all of its essential capabilities on-board. Our robots share the same general architecture and basic hardware, but they have integrated abilities to play different roles (goalkeeper, defender or forward) and utilize different strategies in their behavior. Our philosophy in building these robots is to use the least possible sophistication to make them as robust as possible. In the 1997 RoboCup competition, the Dreamteam played well and won the world championship in the middle-sized robot league.

1. Introduction

The Robocup task is for a team of multiple fast-moving robots to cooperatively play soccer in a dynamic environment [6]. Since teamwork and individual skills are fundamental factors in the performance of a soccer team, Robocup is an excellent test-bed for autonomous agents. For this competition each of the soccer robots (or agents) must have the basic soccer skills -- dribbling, shooting, passing, and recovering the ball from an opponent. Each agent then must use these skills in making complex plays according to the team strategy and the current situation on the field.

An agent must be able to evaluate its position with respect to its teammates and opponents, and then decide whether to wait for a pass, run for the ball, cover an opponent's attack, or go to help a teammate; while at the same time following the rules of the game. In the following sections of this paper we will describe the general architecture for an autonomous robot agent and the team strategy, as well as discuss some of the key issues and challenges in the creation of a team of autonomous soccer agents.

2. General Architecture

For this project we define an autonomous agent as an active physical entity intelligently maneuvering and performing in realistic and challenging surroundings [4]. The critical design feature of our autonomous soccer agents is that each agent has all of the its essential capabilities on-board, so that every robot is self-contained. Autonomous systems, such as these, need to be physically strong, computationally fast, and behaviorally accurate to survive the rapidly changing environment of a soccer field. In our general architecture, great

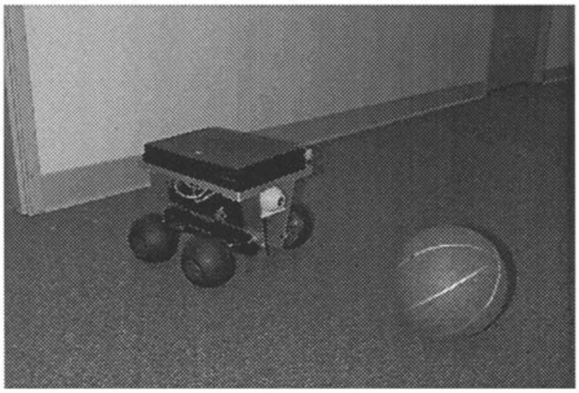

Figure 1. An Autonomous Soccer Robot

importance is given to an individual robot's ability to perform on its own, without outside guidance or help of any kind. Each robot agent bases its behavior on its own sensor data, decision-making software, and eventually communication from other teammates. There are many techniques in robot agent modeling and design [1,2,3,7]; however, much of the work in this area has focused on single agent performing and sometimes under external supervision. We believe that the further of robot agents lies in total autonomous, with the capability of learning and adaptation to the environment [4,5]. Moreover, agents have to be intelligent enough to cooperate among themselves.

Each robot consists of a single 586 based computer on a 30cm x 50cm, 4-wheel drive, DC model car (Figure 1). The computer can run a program from its floppy drive or hard drive. However, due to the robot's very harsh working environment, we decided to store our program in a Solid State Disk (SSD) to avoid any mass-storage malfunction during competition. The robots can be trained and the programs can be upgraded by attaching a floppy drive, a hard drive or by using the on-board networking facility.

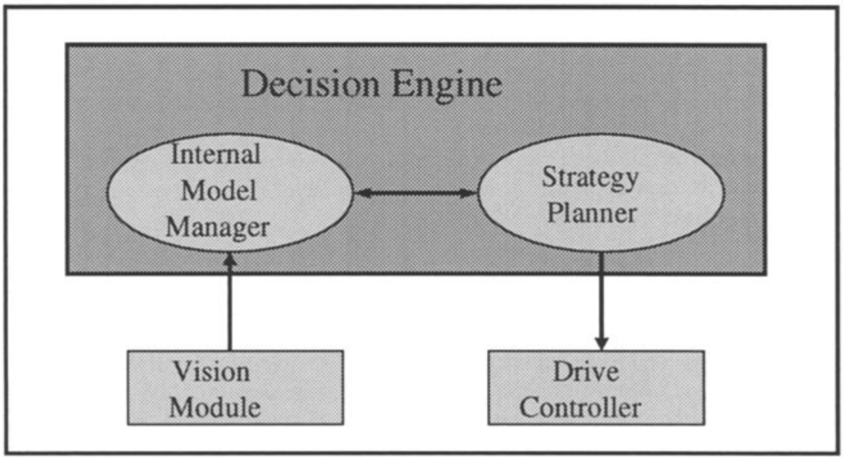

Figure 2. Diagram of the general architecture

The inputs from the on-board camera are collected through a parallel port. Every agent uses the input from the sensors as well as an internal model of the environment and itself to decide its next action. After determining the proper action the robot is steered by commands sent through the I/O ports to the motor controller. Two DC motors, one on each side, provide full motion by quickly turning left or right, and by moving forward or in reverse. Each robot is designed in a modular way, so that we can easily add new software or hardware to extend its working capabilities.

The three main software components of a robot agent are the vision module, decision engine, and drive controller. A diagram showing the interactions between these different modules are shown in Figure 2. The vision module outputs a vector for every frame taken by the agent's on-board camera. Each vector contains the positions of the objects in the frame, such as the ball, players and the goal. This information is then processed by the decision engine. The decision engine is composed of two processing units - the internal model manager and the strategy planner. The model manager takes the vision module's output vectors and maintains an internal representation of the key objects in the soccer field. The strategy planner combines the information that is contained in the internal model with its own strategy knowledge, in order to decide the robot's next action. Once the action has been decided, the command is sent to the drive controller that is in charge of properly executing the command. In the following sections we explain each component in further detail.

```
xx:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::xxxxx::
xx::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::xxxxxx::
xx:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::xxxxx:::
x:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::xxxxxxxxx
-x:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::xxxxxxxxx-------------
-xx::::::::::::::::::::::::::::##########xxxxx------------------------
-xx::::::::::::xxx###################-------------------------
-xxxxxxxxxxxxx###############----------------------------
-xxxxxxx--------################-----------------------------
---------------##################--------------**********----
-----------------###--#######-------------------**************__
-------------------------####-----------------------***************_
-------------------------------------------------------------*************__
--------------------------------------------------------------**********____
------------------------------------------------------------------------------
------------------------------------------------------------------------------
```

Figure 3. Internal representation of the visual input from Figure 1
(Numeric color values are represented here as ASCII characters for a better understanding of the picture.)

3. Vision Module

Just as eyesight is essential to a human player, the visual input to a robot is critical for decision making in a soccer game. A robot relies on its visual input to determine its current position on the field, the positions of other players, the ball, the goals, and the field lines (sidelines, end of field, and penalty area). Each soccer agent receives visual input from an on-board camera. Due to its simplicity, robustness, and software availability, we have chosen Connectix Color PC QuickCam [8] as the on-board visual input hardware. This is a CCD-based camera that outputs 658x496 RGB pixels through the parallel port of the agent's computer. The camera also contains a micro-controller and other electronics to control operation of the camera and pixel data transmission.

To be able to navigate and perform a game plan, our agents depend on visual cues like relative position of the goals, the ball, and other agents. Thus, the agents need to recognize these objects through their visual systems. The output from the QuickCam consists of a series of 3-byte RGB pixels, with each byte representing the intensity of each corresponding color (0 = color absent, 255 = maximum color intensity). The vision software checks each pixel color to see if it may belong to one of the key objects on the field. The pixels are then organized as color clusters to be mapped to identifiable objects. The size and position of each color cluster are used to calculate direction and distance from the actual object on the field. Direction is calculated as an offset of the perceived object with respect to the center of the screen, and distance results from comparing the perceived object size and position within the frame with the known real object size. An example of such an internal representation of objects is shown in Figure 3.

The limitations of the on-board computing power, and the need for a fast, real-time reactive behavior, cooperate to constrain the level of feasible complexity in the algorithms to be run by the agents. Based on the current hardware configuration, visual frames are acquired at a minimum rate of 10 per second. If the processing of a given frame takes more than 0.1 second, the latest frame is processed next, and any intermediate buffered frames are discarded. In this way agents overcome possible lack of synchronization due to variable time needed to process a given frame. It is important to point out that before the agents can correctly recognize objects on the field, diverse lighting conditions and opponent colors mandate an adjustment of the comparison intensity values for each colored object.

2.1. A Vision Calibration Tool

We have developed a calibration tool that can automatically adjust the parameters of the vision system to account for the changes in the robot's surroundings. This vision tool gives the robot the capability to adapt to the new environment. Our adaptive color tracking system uses a supervised learning system based on the inductive learning.

In the first step we take several pictures of the different objects to be recognized in the new environment. The system is attempting to learn the correspondence between the colors and the objects. We repeat the procedure for a given object with a different angle and shadow. In the second step the adaptive color tracking system finds a set of rules to recognize each object, distinguishing which set or class of RGB pixels are associated with a specific object. As the number of colors increase by entering different objects on a game field, the learned rules become more complicated.

RED	GREEN	BLUE	CLASS
240	115	78	Ball
195	95	62	Ball
63	185	90	Field
58	193	115	Field
128	104	213	Goal

Figure 4. Object classification based on RGB

The input for each rule is a combination of Red, Green and Blue intensity for a given pixel in a frame. The output of a rule (the right hand side) would be a class which could be Ball, Wall, Opponent-player, Team-mate, Own-goal and Opponent-goal. As each of these items has a different color in the field. Figure 4 shows an example of input data and a simple rule to differentiate between Red (Ball), Green (Field) and Blue (Goal).

One of the common methods in knowledge discovery and intelligent data analysis is induction. Tree induction methods produce decision trees, concerning the data set as a result of their classification process. To find a valid interpretation in our large frame database we can employed any decision tree algorithm to find the needed association rules between the frame color patterns and the class of objects. In this approach we applied C4.5 [9] as the main induction method. C4.5 package generates decision trees, which provides a classification for every object within the database. The package allows the decision tree to be simplified, using a pruning technique which reduces the size of the tree according to a user-defined confidence level.

Once the vision system has been calibrated to the new environment, it can then process the necessary information about the objects that are perceived in the camera frames. This information (the size and position of the object in the frame) is then sent in the form of object or frame vectors to the decision engine, where it is used to decided the actions of the robot.

3. Decision Engine

This component makes decisions about the actions of an agent. It receives input from the sensor modules and sends move commands to the drive controller. The decision engine bases its decisions on a combination of the received sensor input, the agent's internal model of its environment, and knowledge about the agent's strategies and goals. The agent's internal model and strategies are influenced by the role the agent plays on the soccer field. There are three types of agent roles or playing positions: goalkeeper, defender, and forward. The team strategy is broken down into pieces and instilled in the role strategies of each of the agents. Depending on the role type, an agent can be more concerned about a particular area or object on the soccer field, e.g. a goalkeeper is more concerned about its own goal, while the forward is interested in the opponent's goal. These differences are encoded into the two modules that deal with the internal model and the agent's strategies.

Together, the internal model manager and the strategy planner, form the decision engine. These sub components of the decision engine communicate with each other to formulate the best decision for the agent's next action. The model manager converts the vision module's frame vectors into a map of the agent's current environment, as well as generating a set of object movement predictions. It calculates the salient features and then communicates them to the strategy planner. To calculate the best action, the strategy planner uses both the information from the model manager and the strategy knowledge that it has about the agent's role on the field. It then sends this information to the drive controller and back to the model manager, so that the internal model can be properly updated.

3.1. Model Manager

Because the soccer agents need to know about their environment and themselves before taking an appropriate action, an internal model is used to address this problem. The model manager is responsible for building and maintaining the internal model that provides this information. The internal model contains a map of the soccer field and location vectors for nearby objects. The object vectors from the vision module are used to create the location vectors for the internal model.

A location vector consists of four calculations or elements; distance and direction to the object and the change in distance and direction for the object. The change in distance and direction is used for predicting the object's new location. The model manager continuously receives frame vectors from the vision module and updates the location vectors accordingly.

The model manager keeps track of information based on the role of the robot on the soccer field. For a goalkeeper the information will include the location vectors for the goal, ball and opponent's forwards. This is necessary for not overloading the strategy planner with extra information.

An internal model is necessary for several reasons. Since the visual information provided by the vision module is incomplete, the robot can see only the objects that are within its current visual frame. More information about the environment can be deduced by using a map of the soccer field and the historical data. For example, if the robot can see and calculate the distance to the left wall of the field, it can find out the distance to the right wall even when the right wall is not visible. Similarly, the approximate location of an opponent that was previously visible, but currently not in view can be calculated using the historical data from

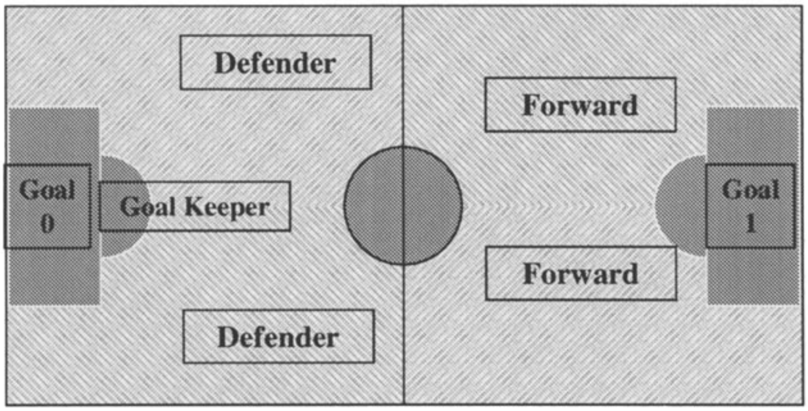

Figure 5. Playing positions of agents on the Soccer field

the previous frames. This also provides greater robustness for the robot. If the camera fails for a few cycles (e.g. due to a hit or being blocked etc.), the robot can still operate using its internal model of the environment.

An internal model is also necessary for predicting the environment. This is important for the strategy planner. For example, to intercept the ball the robot needs the current location of the ball and also a prediction of the current heading of the ball, so that it can calculate an intercept course to the ball. The internal model is also used for providing feedback to the strategy planner to enhance and correct its actions. The strategy planner tells the model manager what actions will be taken and the model manager updates the internal model using this information. It then receives information from the vision module and compares it with the expectations contained in the model. Any discrepancies are reported to the strategy planner, so that it can then use this information to fine tune its operations.

3.2. Strategy Planner

In order to play a successfully soccer game, each robot must react appropriately to different situations in the field. This is accomplished by the strategy planner that resides in the decision engine on each robot. Since a strategy (or a policy) is a mapping from situations to actions, let us first define situations and actions for our robots. As we mentioned in the description of model manager, objects in the soccer field are represented by their relative positions to the observer. Internally, an object is represented by a location vector of four elements: the distance between the object and the observer, the direction of the object, and change in distance and direction. The values of the first two elements are qualitative: the distance can have values near, medium, far, and ? (unknown). The direction can have values left, center, right, and ? (unknown). To increase the response time of our decision process, not all objects in the field are constantly tracked, instead only those that are currently in the visual field of the observer. (Those objects that are currently not in view have ? as their values in the location vector.)

Based on this terminology, we define a situation as a set of observed location vectors in the field. For example, if a forward player is facing the opponent's goal and between the goal and the ball (see Figure 5), it is represented as the following vector:

{ **Ball: <?, ?, ?, ?>,**
Goal0: <5, 11, ?, ?>,
... }

The basic actions of robots are the five commands to the motors; they are move-forward, move-backward, stop, turn-left and turn-right. Based on these basic actions, a set of compound actions or behaviors is then defined. The set of compound actions includes kick, line-up, intercept, homing, and detour. Some brief descriptions of these actions are shown in Figure 6.

KICK: kick the ball
LINE-UP: move to line up the ball and the goal.
INTERCEPT: calculate intercept path to the ball
HOMING: go back to its "home" position
DETOUR: go around the ball

Figure 6. Examples of compound action

Each action has a termination condition and a time threshold. For example, the termination condition of line-up is when the robot is behind the ball and the goal, the ball, and itself are on the same line. In case actions are not successfully terminated, they will time-out as soon as the duration of the action passes the time threshold.

Based on the defined situations and actions, the task of strategy planner is to select the right action for a given situation. This decision process is captured by a Policy Table shown in Figure 7. As we can see from this table, each line is a rule that maps a situation to an action. For example, if the position of ball is unknown in the current situation, the action is to turn left for 30 degree in search of the ball.

Situation					Action
Ball	Goal0	Goal1	Wall	Player	X
<?,?>	<.,.>	<.,.>	<.,.>	<.,.>	Turn Left(30)
<x, y>	<.,.>	<.,.>	<.,.>	<.,.>	LINE-UP
....	...	...	...	...	...

Figure 7. A Policy Table

There five positions a robot can play on our soccer team: left-forward, right-forward, left-defender, right-defender, and goalkeeper. Shown in Figure 5, each player has its own territory and home position. For example, the left-forward has the territory of the left-forward quarter of the field, and its home position is near the center line and roughly 1.5 meter from the left board line. (The territory of the home position for the right-forward is symmetric to that of the left-forward). Similarly, the left-defender is in charge of the left-back quarter of the field and its home position is at the left front of the base goal. Given this configuration, we now describe the policy tables for each of the three types of robots: goalkeeper, forward and defender.

Goalkeeper

For the goalkeeper, the two most important objects in the field are the ball and its own goal. Its home position is in front of the goal, and its policy is to keep itself in the line of the ball and the goal. Since most of its actions are parallel to the base line, the goalkeeper's camera is mounted on the side (for all other robots, the camera is mounted in the front), so that it can move sideways while keeping an eye on the ball. Its policy table is as follows:

Situation →	Action
Ball =<near, left>, Goal0=<near,center>	Move-left
Ball =<near, right>, Goal0=<near,center>	Move-right
Ball =<far, _>	Homing
Ball =<?,_>, Goal0=<near,left>	Turn-right
Ball =<?,_>, Goal0=<near,right>	Turn-left
......	......

Figure 8. The *goalkeeper* policy table

As we can see, the first two actions are to prevent the ball from entering the goal, the third action is to position itself, and the last two actions are to look for the ball. Note that although policy tables are easy to describe conceptually, their implementation in the real system require much engineering to make them correct and robust.

Forward

The talks of forward is to put the ball into opponent's goal whenever possible. Like the goalkeeper, it must look for the ball when it is not in sight, and head back to its home position when the ball is out of its territory. The main difference is that the forward's main interest is to kick the ball towards the opponent's goal (Goal1 in our current example) whenever it can. So its policy table, shown in Figure 9, must reflect that.

Note that as long as the ball and Goal1 are both in sight, the forward will kick the ball. The policy also tells the robot to search for the ball whenever it is not in sight (e.g., seeing only the wall). It returns to its home position if it sees the opponent goal but not the ball. Whenever it sees the ball and its own goal (Goal0), it must make a detour, so that it can kick the ball in the correct direction.

Situation	→ Action
Ball =<near, center>, Goal1=<.,~?>	Kick
Ball =<?, _>,Goal1=<.,~?>	Homing
Ball =<?, _>,Goal1=<_,left>	Turn-right
Ball =<?, _>,Goal1=<_,right>	Turn-left
Ball =<?, _>,Wall =<near,?>	Turn-left
Ball =<near, _>,Goal0=<far,?>	Detour
......	......

Figure 9. The *forward* policy table

Defender

The defender's policy is very similar to that of the forward, except that the distance to Goal1 is further away compared to the position of the forward. Similar to the goalkeeper, it also tries to position itself between the ball and its own goal (Goal0).

4. Drive Controller

As mentioned before, each robot consists of a single 586 based computer on a 30cm x 50cm, 4-wheel drive, DC model car (Figure 1). The inputs from the on-board camera are collected through parallel and serial ports. The drive controller takes commands from the decision engine, and steers the robots through its I/O ports. Four I/O ports are connected to the DC motor drivers and steer the robot by quickly turning left or right, and by moving forward or in reverse. The drive controller stops the motion by disconnecting the move order. The car also has the capability to spin in place by turning the two sides of wheel into opposite direction. In order to make movements that are as precise as possible, the effects of actions are controlled by the amount of time of a command. To deal with much of the uncertainty associated with the system, closed-loop control in software is used whenever possible.

5. Related Work

The current research follows the original work from a prediction-based architecture called LIVE [5] for integrating learning, planning and action in autonomous systems [4], and a behavior-based robot control system [1,2]. Compared to similar architectures in recent agent literature (for example [7]), our approach uses the closed loop control theory in many aspects of robot building, and uses the internal model to help detect and recover from errors. Moreover, our robot agent can also be configured quickly into different soccer playing roles and that has greatly increased the flexibility of the entire robot team.
Our philosophy on multi-agent collaboration [6] is that if each agent has a sufficient understanding of other agents' action, then collaboration can be accomplished without any explicit communication. In our team, there is an implicit form of collaboration in that each agent has a particular role in which it knows its function and behaviors, as well as those of its teammates. Indeed, some limited collaboration behavior has been observed during the competition, and additional evidence of this hypothesis can also be found in the description of the champion team in the simulated league in this volume.

6. Conclusion and Future Work

Conventionally, agents in the real world are controlled and guided by an external supervisor when performing complex tasks. In this study, we propose an autonomous system model for soccer robots that do not use any external computation resources, information, or guidance. Our soccer agents are designed with simple structures, but demonstrate that with an acceptable strategy and system modularity they can achieve the goal of autonomous behavior in a context of teamwork. These autonomous soccer robots use on-board cameras as their own sensors, and decide appropriate actions based on-board computers. Three main features seem to contribute the most to this success. First, the model manager is responsible for translating and interpreting the vision information in order to obtain the knowledge necessary for the strategy planner. Second, the closed-looped control mechanism deals with many intrinsic uncertainties in the soccer domain. Third, the modularity of hardware and software design of the system has made it possible for a single architecture to play multiple roles.

In our future work, we would to improve the quality of autonomous sensing and acting, as well as extend the collaborations between robots. In particular, we would like to add more and heterogeneous sensors, such as sonar and others to speed up an agent's decision making and increase their efficiency and accuracy. We would also like to extend our system to use explicit communication for agent collaboration.

7. Acknowledgements

We would like to thank USC/ISI, especially people in the Division of Intelligent System, for their generous moral and financial support for the Dreamteam. We are also grateful for the international travel support provided by Japan Airline, and an Omnibook 600 laptop computer provided by Hewlett Packard for earlier experiments on the vision and control systems. Finally, thanks to Patrick Reynolds from the University of Virginia for the free software package *cqcam* for using QuickCam.

References

1. Arbib, M. Perceptual Structures and Distributed Motor Control. In *Handbook of Physiology -- The Nervous System*, II, ed. V. B. Brooks. American Physiological Society, 1981.
2. Arkin, R.C. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research* (1987) 92-112.
3. Brooks, R. A. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1), 1986.
4. Shen, W.M. *Autonomous Learning From Environment*. Computer Science Press, W.H. Freeman, New York. 1994.
5. Shen, W. M. LIVE: An Architecture for Autonomous Learning from the Environment *ACM SIGART Bulletin (Special Issue on Integrated Cognitive Architectures)* 2(4), 151-155. 1991.
6. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.. Robocup: The Robot World Cup Initiative. In *Proceeding of the first International Conference on Autonomous Agents*. Marina del Rey, CA, 1997.
7. Garcia-Alegre, M.C., Recio, F.. Basic Agents for Visual/Motor Coordination of a Mobile Robot. In *Proceeding of the first International Conference on Autonomous Agents*. Marina del Rey, CA, 1997.
8. Connectix Corporation, Connectix Color QuickCam, May 1996.
9. Quinlan, J. R.. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.