A new longer version for AI Magazine, Spring 1998.

Towards Integrated Soccer Robots

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

Abstract

Robot soccer competition provides an excellent opportunity for integrated robotics research. In particular, robot players in a soccer game must recognize and track objects in real-time, navigate in a dynamic field, collaborate with teammates, and strike the ball in the correct direction. All these tasks demand robots that are autonomous (sensing, thinking, and acting as independent creatures), efficient (functioning under time and resource constraints), cooperative (collaborating with each other to accomplish tasks that are beyond individual's capabilities), and intelligent (reasoning and planing actions and perhaps learning from experience). Furthermore, all these capabilities must be integrated into a single and complete system, and this raises a set of challenges that are new to individual research disciplines. This paper describes our experience (problems and solutions) in these aspects. 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 sophistication to make them as robust and integrated as possible. In RoboCup97, these integrated robots performed well and our Dreamteam won the world championship in the middle-sized robot league.

1. Introduction

The RoboCup task is for a team of fast-moving robots to cooperatively play soccer in a dynamic environment [5,7]. Since individual skills and teamwork are fundamental factors in the performance of a soccer team, Robocup is an excellent test-bed for integrated robots. Each soccer robot must have the basic soccer skills— dribbling, shooting, passing, and recovering the ball from an opponent, and must use these skills to make complex plays according to the team strategy and the current situation on the field. For example, depending on the role it is playing, an agent must 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.

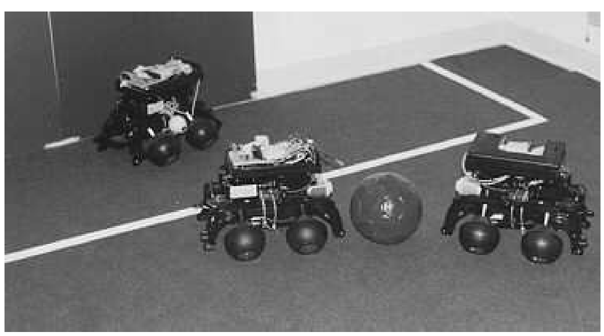

Figure 1: Integrated Soccer Robots

To build agents with soccer-playing capabilities, we must design an architecture to integrate hardware and software and balance between the system's performance, flexibility and resource consumption. Within this architecture, we must have (1) a fast and reliable vision component to detect various static and dynamic objects and to adapt to different lighting conditions and color schema., (2) an effective and accurate motor system to deal with uncertainties in motor control, and (3) a set of software strategy for robots to play different roles to increase the flexibility of the team. Since solutions to these tasks require integration of several distinct research fields, such as robotics, AI, vision, etc., we have to address some of the integration problems that have not been attacked before. For example, since our robots perceive and process all visual images on-board, the noise ratio in the perception is higher than those that use static global vision systems. This demands some extra cautions in the way how the pictures are taken. Furthermore, since the environment is highly dynamic, uncertainties associated with the motor system will vary with different actions and with the changes of power supply. This posts additional challenges on real-time reasoning about action in comparison with systems that are not integrated as complete and independent physical entities.

In the following sections of this paper, we will address the above tasks and problems in detail. The discussion will be organized as descriptions of component in our systems, with highlights on key issues and challenges. The related work will be discussed at the end.

2. The System Architecture

Our design philosophy for the system architecture is that we view each robot as a complete and active physical system that can intelligently maneuver and perform challenging tasks in a realistic environment. In order to survive the rapidly changing environment in a soccer game, each robot must be physically strong, computationally fast, and behaviorally accurate. Considerable importance is given to individual robot's ability to perform on its own without any off-board resources such as global, birds-eye view cameras or remote computing processors. Each robot's behavior must base on its own sensor data, decision-making software, and eventually communication with teammates.

The hardware configuration of our robot is as follows (see examples in Figure 1). The basis of each robot is a 30x50cm, 4-wheel, 2x4 drive, DC model car. The wheels on each side can be controlled independently to make the car spin fast and maneuver easily. The two motors are controlled by the on-board computer through two serial ports. The hardware interface between the serial ports and the motor control circuits on the vehicle are designed and built by ourselves. The robot can be controlled to move forward and backward, and turn left and right. The "eye" of the robot is a commercial digital color camera called QuickCam made by Connectix Corp.. The images from this camera are sent into the on-board computer through a parallel port. The on-board computer is an all-in-one 133MHz 586 CPU board extensible to connect various I/O devices. There are two batteries on board, one for the motor and the other for the computer and camera.

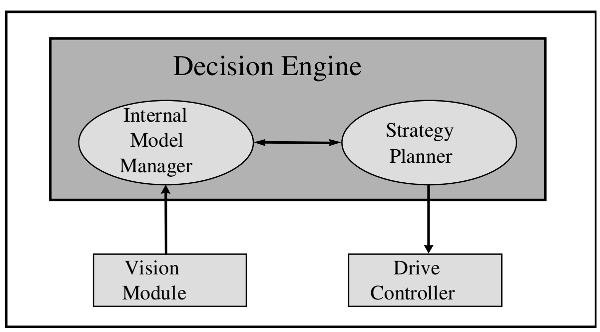

Figure 2: The System Architecture

The software architecture of our robot is illustrated in Figure 2. The three main software components of a robot agent are the vision module, the decision engine, and the drive controller. The task of the vision module is to drive the camera to take pictures, and to extract information from the current picture. Such information contains an object's type, direction, and distance. This information is then processed by the decision engine, which is composed of two processing units - the internal model manager and the strategy planner. The model manager takes the vision module's output and maintains an internal representation of the key objects in the soccer field. The strategy planner combines the internal model with its own strategy knowledge, and decides the robot's next action. Once the action has been decided, a command is sent to the drive controller that is in charge of properly executing. Notice that in this architecture, the functionality is designed in a modular way so that we can easily add new software or hardware to extend the system's working capabilities.

We use Linux as the on-board operating system and built a special kernel with 4MB file system, all compressed on a single 1.4MB floppy disk for easy down-loading. The entire software system (for vision, decision, and motor drive) consists of about 6,500 lines of C and C++ code.

One challenge we faced during the design of architecture was to draw a proper line between hardware and software. For example, to control the two motors, we had a choice between using one serial port (a commercial laptop) or two serial ports (a complete all-in-one CPU board), we chose the later because we decide to solve the interface issue completely in hardware. (The former requires a complex software protocol and hardware interface). In retrospect, it seems that our decision on this issue and other issues in architecture was mainly driven by two factors: feasibility and robustness.

3. The Vision Module

Just as eyesight is essential to a human player, a soccer robot depends almost entirely on its visual input to perform its tasks, such as determining the direction and distance of objects in the visual field. These objects include the ball, the goals, other players, and the lines in the field (sidelines, end of field, and penalty area). All this information is extracted from an image of 658x496 RGB pixels, received from the on-board camera via a set of basic routines from a free package called CQCAM, provided by Patrick Reynolds from the University of Virginia.

Since the on-board computing resources for an integrated robot are very limited, it is a challenge to design and implement a vision system that is fast and reliable. In order to make the recognition procedure fast, we have developed a sample-based, active method that can quickly focus attention on certain objects. Depending on the object that needs to be identified, this method will automatically select certain number of rows or columns in an area of the frame where the object is most likely to be located. For example, to search for a ball in a frame, this method will selectively search only a few horizontal rows in the lower part of the frame. If some of these rows contain segments that are red, then the program will report the existence of the ball (recall that the ball is painted red). Notice that domain knowledge about soccer is useful here to determine where and how the sample pixels should be searched. For example, since the ball is often on the floor, only the lower part of the image needs to be searched when we are looking for the ball. Similarly, when the robot is looking for a goal, it will selectively search columns across the image and the search should from the floor up. Using this method, the speed to reliably detect and identify objects, including take the pictures, is greatly improved; we have reached frame rates of up to 6 images per second.

To further increase the speed of perception, the above vision routine is used to facilitate the focus of attention on important objects. Thus, instead of searching for all objects all the time, the system will first look for the ball. Depending on the results, the vision will selectively look for other objects. For example, if the ball is not found, then no other objects need to be searched. If the ball is found, then next important object should be goals. This way, vision is highly active and selective.

To increase the reliability of object recognition, the above method is combined with two additional processes. One is the conversion of RGB to HSV, and the other is "neighborhood checking" to determine the color of pixels. The reason we convert RGB to HSV is that HSV is much more stable than RGB when light conditions are slightly changed. Neighborhood checking is an effective way to deal with noisy pixels when determining colors. The basic idea is that pixels are not examined individually for their colors, but rather grouped together into segment windows and using a majority-vote scheme to determine the color of a window. For example, if the window size for red is 5 and the voting threshold is 3/5, then a line segment of "rrgrr" (where r is red and g is not red) will still be judged as red.

Object's direction and distance are calculated based on their relative position and size in the image. This is possible because the size of ball, goal, wall, and others are known to the robot at the outset. For example, if one image contains a blue rectangle of size 40x10 pixels (for width and height) centered at x=100 and y=90 in the image, then we can conclude that the blue goal is currently at 10 degree left and 70 inches away.

To make this vision approach more easily adjustable when environment is changed, we have kept the parameters for all objects in a table, in a separate file. This table contains the values of camera parameters such as brightness and contrast, as well as window size, voting threshold, average HSV values, and search fashion (direction, steps, and area). When the environment or the vision task is changed, only this file needs to be changed and the vision program will function properly.

Given the speed of current processing rate of object recognition, it is now possible to track the moving direction of the ball and other players. To do so, a robot will take two consecutive pictures, and compare the locations of the ball in these two pictures. If the direction of the ball moves to left (right), then the robot concludes that the real ball is moving towards left (right). In fact, this is how our goalkeeper predicts the movement of an incoming ball.

Vision modules such as the one described here also face problems that are unique for integrated robots. For example, images will have much higher noise-ratio if the robot is not careful about when and how the pictures are taken. It took us quite a long time to realize this problem. At first, we were very puzzled by the fact that although the vision system is tested well statically, our robot would sometimes behave very strangely as if it is blind. After many trials and errors, we noticed that pictures that are taken while the robot is fast moving have very low quality. Such pictures are not useful at all in decision-making. Since then, special care has been given to the entire software system and pictures are taken only when the robot is not moving.

4. Drive Controller

As specified in the system architecture, the drive controller takes commands from the decision engine, and sends the control signals to the two motors in parallel via two serial ports and a special-purpose hardware interface board. The interface provides a bridge between the two systems (the computer and the robot body) that have different power supplies.

Since the two motors (one for each side of the robot) can be controlled separately, the robot can respond to a large set of flexible commands. The basic ones include turning left and right, moving forward and backward. Others include making a big circle in the forward-left, forward-right, back-left and back-right direction. This is done by giving different amounts of drive force to the different sides. In the competition, however, we only used the basic actions for reliability reasons.

One challenge for building this simple drive controller is how to make the measured movements, such as moving forward 10 inches or turning left 35 degree. We solve this problem first by building a software mapping from the measurements of movement to the time duration of the motor running. For example, a command turning left for 30 degree would be translated by this mapping to forwarding the right-motor and backwarding the left-motor for 300ms. This solution works well when all components in the system, especially the batteries, are in perfect condition and floor material is good for wheel movement. But the accuracy of this open-loop control "deteriorates" when the power decreases or as the environment changes. Once this happens, the whole robot will behave strangely because the motor movements are no longer agreeing with the control signals.

To solve this problem, we have made all motor controls closed-loop in the entire system. Instead of saying "turning 75 degree," we also specify the termination criteria for such a turn command. For example, if the purpose of this turning is to find a goal, then the program will repeat issue smaller turnings until the goal is found. With the closed-loop control commands, the reliability of motor control has increased considerably and become more robust with respect to power fluctuation.

This closed-loop motor control also results in one of our secret weapons for well-behaved dribbling actions. Different from other team's dribbling action which may quickly lose the ball, our robot uses closed-loop control and continuously adjusts its moving direction according to the current direction of the ball. This approach worked very well in the competition, and contributed a great deal to the success of our team.

5. The Decision Engine

Based on the existing theories of autonomous agents (for example [9]), integrated robots are best to be model-driven. This principle has guided our design and implementation of the brain of our robots, namely the Decision Engine. Compared to other model-less and pure-reactive approaches, our approach could in principle demonstrate more intelligent behaviors without sacrificing the ability to quickly react to different situations.

As one can see in Figure 2, the Decision Engine receives input from the vision module 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 distributed into the role strategies of each individual agent. Depending on the role type, an agent can be more concerned about a particular area or object on the soccer field. For example, a goalkeeper is more concerned about its own goal, while a forward is more 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.

The decision engine consists of two sub-modules: the internal model manager and the strategy planner. These sub-modules communicate with each other to select the best decision for the agent's next action. The model manager converts the vision module's output into a "map" of the agent's current environment, as well as generating a set of object movement predictions. It calculates the salient features in the field 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.

5.1. Model Manager

For robots to know about their environment and themselves, the model manager uses the information detected by the vision module to construct or update an internal model. This model contains a map of the soccer field and location vectors for nearby objects.

A location vector consists of four basic elements; distance and direction to the object and the changes in distance and direction of the object. The changes in distance and direction are used to predict a dynamic object's movement; these are irrelevant for objects that are static. Depending on the role a robot is playing, the model manager actively calls the vision module to get the information that is important to the robot and updates the internal model. For example, if the robot is playing goalkeeper, then it needs to know constantly about the ball, the goal, and its current location relative to the goal.

An internal model is necessary for several reasons. First, since a robot can see only the objects within its current visual frame, a model is needed to keep information that is perceived previously. For example, a forward robot may not able to see the goal all the time. But when it sees the ball, it must decide quickly which direction to kick. The information in the model can facilitate such decision readily. Second, the internal model adds robustness for a 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. Third, the model is necessary for predicting the environment. For example, a robot needs to predict the movement of the ball in order to intercept it. This prediction can be computed by comparing the ball's current direction with its previous one. Fourth, the internal model can be used to provide feedback to the strategy planner to enhance and correct its actions. For example, in order to perform a turn-to-find-the-ball using the closed-loop control discussed above, the internal model provides the determination criteria to be checked with the current visual information.

5.2. 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. Internally, a situation is represented as a vector of visual clues such as the relative direction and distance to the ball, goals, and other players. A strategy is then a set of mappings from situations to actions. For example, if a forward player is facing the opponent's goal and sees the ball, then there is a mapping to tell it to perform the kick action.

For our robots, there are five basic actions: forward, backward, stop, turn-left and turn-right. These actions can be composed to form macro actions such as kick, line-up, intercept, homing, and detour. For example, a detour action is basically a sequence of actions to turn away from the ball, move forward to pass the ball, turn back to find the ball again, and then forward to push the ball. These compound actions represent a form of simple planning, and that contributes many of the intelligent behaviors demonstrated by our robots. Indeed, during the competition, the audience cheered when they saw one of our robots made a detour in order to protect our goal.

5.3. Role Specifications

There are five roles that a robot can play for its team: left-forward, right-forward, left-defender, right- defender, and goalkeeper. Each role is actually implemented as a set of mappings from situations to actions, as described above. Each role 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. 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. The mappings for these roles are briefly defined as follows.

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 strategy is to keep itself in 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 other robots, the camera is mounted on the front), so that it can move sideways while keeping an eye on the ball. As we mentioned before, the goalkeeper also predicts the movement of an incoming ball in order to fulfill its strategy in time. There are four compound actions for the goalkeeper. Two actions, move to the left or right side, are used to prevent the ball from entering the goal. The third action is to search for the ball, and the fourth one is to position itself in the best location. This last action is most difficult to implement because the goalkeeper must simultaneously track three types of information: the ball, and its own horizontal and vertical offsets with respect to the goal.

The strategy for the forward role is relatively simple. Its task is to push the ball toward the opponent's goal whenever possible. A forward must look for the ball, decide which direction to kick when the ball is found, and perform the kick or detour action appropriately. This strategy proved to be fast and effective in the competition.

The defender's strategy is very similar to that of the forward, except that the distance to the opponent goal is substantially larger compared to the position of the forward. Similar to the goalkeeper, it also tries to position itself between the ball and its own goal. The most difficult action for a defender is to reliably come back to its position after it chases the ball away.

6. Collaboration and Learning

As we can see from the role specifications, there is no explicit collaboration built in the role strategies. Rather, we believe that if every robot plays its role perfectly, then collaboration will naturally emerge. Indeed, during the competition, we saw two of our forwards helped each other to score a goal: one robot rescued the ball from two opponents, and the other robot saw the ball right in front of the goal, and pushed it in. In the future, we will improve our role specification to include passing and assisting ball dribbling, preferably without any explicit communications.

Learning is an important issue yet to be addressed, although our model-based approach provides the basic elements for its implementation. One particular area where learning is especially needed is the calibration of the vision system in a new environment. In the long run, it would also be nice to have the robot learn from its own successes and mistakes (such as scoring at one's own goal).

7. Related Work

The approach we used in RoboCup97 is descended from an earlier, integrated system called LIVE [10] for autonomous learning from the environment [9]. It also shares ideas with integrated cognitive architectures in [6], layered-controlled robots [3], behavior-based robots [1,2], as well as recent progress in Agent research [4]. The unique feature of our robots, however, is the use of internal model and closed-loop control in action planning and execution. Our earlier work along this line includes a silver medal winner robot called YODA [8] in the 1996 AAAI Robot competition for indoor navigation and problem solving.

8. Conclusions and Future Work

In building integrated robots that are autonomous, efficient, collaborative, and intelligent, we have demonstrated a simple but effective approach. At the present, it seems that the most effective approach for soccer robots is to build integrated robots using the least-sophistication to achieve the most robustness. In the future, we will continue our design strategy but improve our robots in the areas of collaboration (passing), faster and more reliable sensing, and learning from experience.

References

[1] Arbib, M. 1981. Perceptual Structures and Distributed Motor Control. In *Handbook of Physiology- The Nervous System, II*, ed. V. B. Brooks, 1449-1465. American Physiological Society.

[2] Arkin, R. C. 1987. Motor Schema-Based Mobile Robot Navigation. *International Journal of Robotics Research*, 92-112

[3] Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1).

[4] Garcia-Alegre, M. C. and F. Recio. 1997. Basic Agents for Visual/Motor Coordination of a Mobile Robot, in *Proceeding of the first International Conference on Autonomous Agents*, , 429-434.

[5] Kitano, H., M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa. Robocup: The Robot World Cup Initiative, in *Proceeding of IJCAI-95 Workshop on Entertainment and AI/Alife*, Montreal, 1995.

[6] Laird, J.E. (ed) Special Issue on Integrated Cognitive Architectures. *ACM SIGART Bulletin* 2(4).

[7]. Mackworth, A and M. Sahota. Can situated robots play soccer? In *Proceeding of. Artificial Intelligence* 94, Banff, AB. May, 1994. Pp 249--254.

[8] Shen, W.M., J. Adibi, B. Cho, G. Kaminka, J. Kim, B. Salemi, and S. Tejada. YODA—The Young Observant Discovery Agent. *AI Magzine*, Spring 1997. 37-45.

[9] Shen, W.M. 1994. *Autonomous Learning From Environment*. W. H. Freeman, Computer Science Press. New York.

[10] Shen, W. M. 1991. LIVE: An Architecture for Autonomous Learning from the Environment. *ACM SIGART Bulletin* 2(4): 151-155.