

---

# The Cornell RoboCup Robot Soccer Team: 1999 – 2003

Raffaello D'Andrea

Sibley School of Mechanical and Aerospace Engineering,  
Cornell University, Ithaca, NY 14853, U.S.A.  
`raff.d'andrea@cornell.edu`

## 1 Introduction

What is RoboCup? Taken directly from the RoboCup Organization web site ([www.robocup.org](http://www.robocup.org)):

RoboCup is an international research and education initiative. It is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined, as well as being used for integrated project-oriented education.

While the scope of RoboCup has grown since its inception in 1997, the main activity remains the robot soccer leagues: Simulation, Small Size, Middle Size, Four-legged, and Humanoid. The Simulation league, as the name implies, does not involve physical hardware, except for computer workstations. The Four-legged league consists of Sony Aibos — robot dogs — programmed to play soccer. The Humanoid league is still in its infancy; the main objective of this league is to demonstrate basic skills, such as standing, walking, and kicking a ball, with a two-legged robot. The remaining two leagues, the Small Size and Middle Size leagues, involve the actual construction of robots that compete in head-to-head soccer matches.

The main difference between the two leagues, contrary to what the names suggest, is the allowed sensing technologies. Teams in the Middle Size league are constrained to use local, on-board vision, while competitors in the Small Size league are permitted to use global vision systems (in addition to local vision systems, if they so desire). As a result, the Middle Size league is dominated by robot localization and perception, while the main challenges in the Small Size league are at the system and integration level. In addition, due to the enhanced situational awareness obtained with global vision, the Small Size league games are much faster and typically involve more team play and collaboration, such as passing.

This article describes the Small Size league Cornell RoboCup team in the time period starting in 1999, the year it first competed, and ending in 2003, after five years of competition. In this time period, the Cornell team won the competition four times and placed third once. See Table 1 for a summary of the competition outcomes since inception or the RoboCup web site for a more detailed summary.

The emphases of this article are on the parts of the system that distinguished the Cornell team from its competitors and led to its success in competition.

In the Small Size league the soccer matches are played by teams composed of up to five robots. Similar to the real game of soccer, the objective is to score more goals than the opponent, subject to well-defined rules and regulations. For example, repeated pushing and charging fouls lead to yellow cards (warnings) and eventual red cards (expulsions). The reader is referred to the RoboCup web site for a detailed list of the rules. The robots and an off-field computer are permitted to communicate via wireless transceivers. A global vision system is typically used for robot and ball position determination. Fig. 1 is a picture from a typical game.



**Fig. 1.** A typical RoboCup game, with the human referee getting ready to blow the whistle and initiate game play

RoboCup is an excellent test bed for developing new tools and techniques for controlling autonomous systems in uncertain and dynamic environments. From an educational perspective, it is also a great means for exposing students to the systems engineering approach for designing, building, managing, and maintaining complex systems [1], [2]. In addition to the RoboCup web site, the reader is referred to [3], [4], [5], [6], and the references therein, for an in-depth description of the competition, its history, and the long-term objectives of RoboCup.

## 2 Physical System Architecture

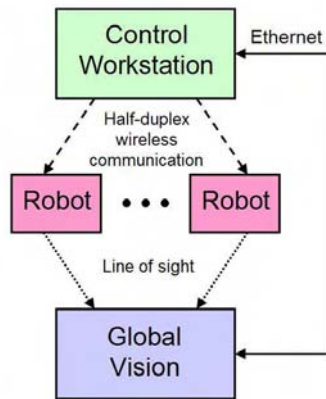
The physical architecture used by the Cornell team is depicted in Fig. 2. A global vision system, with up to three cameras, was used to identify game objects. This information was then fed to the control workstation, whose task was to coordinate the motion of the robots. Various versions of the Windows operating system were

Year	Location	Teams competing	Champion
1997	Nagoya, Japan	4	Carnegie Mellon University
1998	Paris, France	12	Carnegie Mellon University
1999	Stockholm, Sweden	18	Cornell University
2000	Melbourne, Australia	20*	Cornell University
2001	Seattle, USA	20*	Nee Ann Polytechnic
2002	Fukuoka, Japan	20*	Cornell University
2003	Padova, Italy	20*	Cornell University

**Table 1.** Small Size league results.

\*Since 2000, the competition has been restricted to the top 20 teams, as judged by qualification papers and videos.

used. A low-latency version of transmission control (TCP)/Internet protocol (IP) was used for inter-workstation communication. The total round-trip latency of the system was different every year. It constantly decreased as new technology was introduced and better algorithms were developed. In 2003, the total system latency was approximately 80 ms.



**Fig. 2.** Physical architecture

### 3 The Robots

A picture and CAD drawing of a 2003 robot are shown in Fig. 3. Functionally, each robot had the following components: a drive mechanism for locomotion; a dribbling mechanism for imparting back-spin on the ball; a solenoid for kicking and passing the ball; and microprocessor-based electronics for motor control, kick and dribble control, and wireless communication. Some of these are described below.

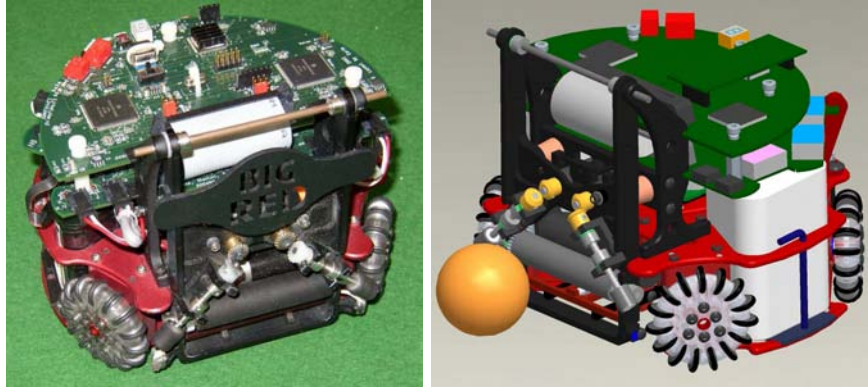


Fig. 3. 2003 robot.

### 3.1 Drive mechanism

Cornell introduced the omni-directional drive for locomotion in 2000, an innovation which was adopted by most other teams. For example, in the 2003 competition, seven out of eight teams in the quarter finals were using omni-directional drives. There are two main advantages for using these drives over their standard two-wheeled counterparts: 1) the robots are more maneuverable, and 2) the problem of trajectory generation is substantially simpler [7]. Contrary to common belief, it is the second advantage that makes omni-directional drive much more superior than two-wheel drive, when viewed from an *overall system* perspective. This is discussed in more detail in Section 4.3.

The maximum *controlled*<sup>1</sup> speeds and accelerations of the robots are listed in Table 2. A rate gyro was introduced in 2002 to substantially increase the performance envelope of the robots.

Year	Max. Speed (m/s)	Max. Accel. (m/s <sup>2</sup> )
1999	1.0	2.0
2000	0.6	1.5
2001	1.0	2.0
2002	1.4	2.5
2003	1.8	3.5

**Table 2.** Evolution of robot performance. Two-wheeled robots were used in 1999, three-wheeled omni-directional robots in 2000 and 2001, four-wheeled omni-directional robots in 2002 and 2003.

<sup>1</sup>The word “controlled” is taken to loosely mean ability to follow a pre-specified trajectory.

### 3.2 On-Board electronics

A different microprocessor was used each year of the competition. The choice was essentially dictated by the expertise of the current team members, and the desire to shift more intelligence onto the robot and away from the control workstation. For example, in 2003, various coordinate transformations and some rudimentary prediction and estimation, made possible by the rate gyro, were implemented directly on the robot.

Wireless communication also varied from year to year. For example, dedicated, low-latency wireless modules in the 433, 418, 869, and 914 MHz bands were used in 2002 and 2003. These wireless modules were used in half-duplex mode, with one transmitter (connected to the control workstation, see Fig. 2) and five receivers (on the robots). In 2003, information was sent to the robots at a rate of 60 times per second, although very little performance degradation was observed at the lower rate of 30 times per second.

## 4 Functional Architecture

The functional architecture is depicted in Fig. 4. There are five main blocks, described in detail below.

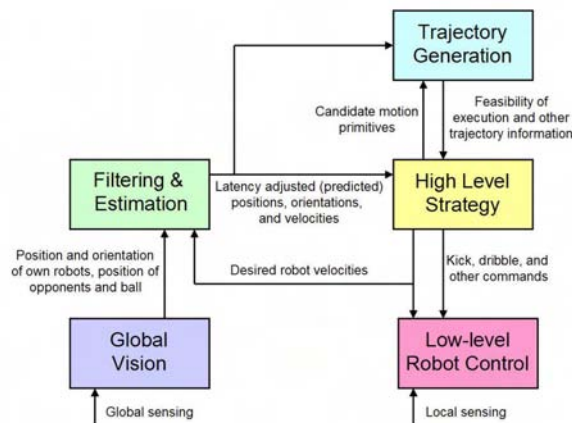


Fig. 4. Functional architecture

### 4.1 Global vision

The global vision system was responsible for extracting the position and orientation of the friendly robots and position of opponents and the ball. While it was possible to extract the orientation of the opponents, this information was of little use due

to the widespread use of omni-directional vehicles and the high rotation speed of two-wheeled vehicles.

Unlike the Middle Size League, computer vision in the Small Size league is relatively straightforward. In particular, all motion is essentially two dimensional (an overhead camera is typically used), and distinct color markers are placed on the tops of the robots. A top view of the 2002 Cornell robots and their vision markers is shown in Fig. 5, as well as the 2002 Vision Calibration Graphical User Interface (GUI).



**Fig. 5.** LEFT: Top view of the 2002 robots. Various color coding schemes were used to make position and orientation determination robust to missed markers. RIGHT: The Vision Calibration GUI. The vision system had to be recalibrated whenever the lighting conditions changed substantially

## 4.2 Filtering and estimation

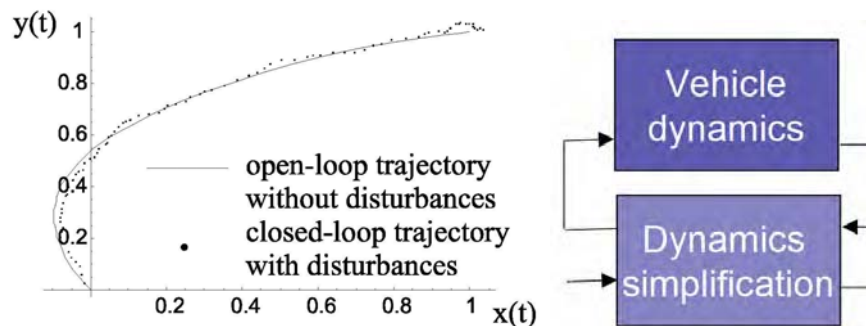
This block was responsible for taking the vision data, extracting velocity information from it, and predicting the location of the robots and the ball in the *robot temporal frame of reference*. In particular, as far as all decision-making blocks were concerned, “now” corresponded to the time when a robot was *observed* to act upon a command. For the opponents and the ball, filtering and estimation/prediction were performed using straightforward Kalman filters. For friendly robots, the commanded robot velocities were also used to obtain a much more accurate prediction of their position. In the absence of disturbances, this essentially nullified the effects of latency. Note that this approach is very similar in spirit to the standard Smith predictor [8], which is often used for time delay compensation.

Various strategies were used to reset the filters when the ball and robots abruptly changed direction (mainly due to collisions), or when substantial discrepancies were observed between the predicted values of the friendly robots and the actual values obtained via computer vision.

### 4.3 Trajectory generation

The responsibility of this block was to take as inputs candidate motion primitives, such as “Move to location (X,Y), as quickly as possible, without hitting anything”, and generating the appropriate robot velocities and other trajectory parameters (time to execute a maneuver, for example).

In the absence of obstacles, nearly time-optimal trajectories were generated for the omni-directional vehicles with very little computational burden: Less than 300 floating-point operations; see [7] for details. A typical trajectory is depicted in the left part of Fig. 6. The basic idea is captured in the right part of Figure 6. A compensator inserted in the vehicle control loop altered the effective dynamics in a way that greatly simplified the associated optimal control problem for trajectory generation. This simplification was achieved with a very small sacrifice in performance [7].



**Fig. 6.** LEFT: Typical result of trajectory generation. RIGHT: Dynamics simplification

The resulting trajectories were then used as primitives for various obstacle avoidance algorithms, such as the randomized path planning algorithms in [9].

Using standard computing platforms, one could thus perform more than one million trajectory calculations per second. Referring to the diagram of Fig. 4, the implications should be clear: the High Level Strategy block was thus free to explore a very large number of nearly optimal motion scenarios.

### 4.4 Low level robot control

This block, which resided on the robots, essentially performed velocity tracking and supervised the control of all other mechanisms on the robot, such as dribbling and kicking. As an example, the block diagram of the inner loop used for angular velocity regulation is found in Fig. 7.

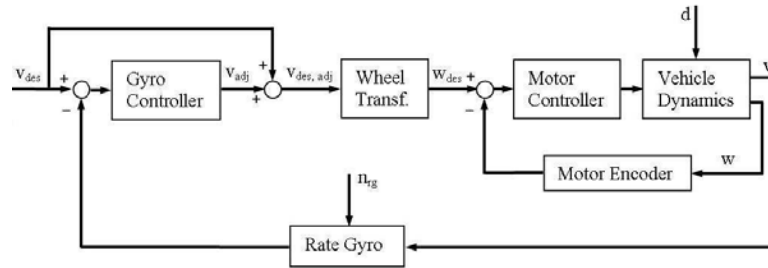


Fig. 7. Angular velocity regulation

#### 4.5 High level strategy

Not surprisingly, this block relied the most on human input for design (such as hard-coded successful sport strategies,<sup>2</sup> or scouting the opponents and altering the behavior of the system before game time) and it was the least amenable to systematic design and analysis. This, of course, is not to say that it is a trivial task to design, test, and implement it. Rather, the system was architected so that all the hierarchical levels below it had solid analytical backbones (such as optimal control for trajectory generation and Kalman filtering for estimation), while most of the heuristics were confined to the High Level Strategy block. This block also changed the most from year to year, as we tried new methodologies and approaches.

In 1999, our first year of competition, we adopted a role-based system similar to what was used by the defending 1998 champion, Carnegie Mellon University [10]. This approach was successful and yielded acceptable performance, but it was very difficult to debug and build upon. In particular, role interdependencies made systematic design extremely difficult; it was often impossible to predict the effects of adding a new role to the system. Having said that, this variability and unpredictability could have its benefits if harnessed by evolutionary optimization techniques.

Fig. 8 captures a simplified version of the High Level Strategy used in 2003. The directed graph in the left part of the figure captures the possible Game States and the transitions between them. One possible Game State, “Opportunistic Offense”, is depicted in the right part of the figure. In a given Game State, each robot was assigned a role. Unlike a pure role-based system, the five roles for each Game State were fixed. Each role, in turn, could make use of various skills. Roles could be reused, as could skills. Both roles and skills were typically parameterized, which resulted in small and manageable sets of roles and skills.

One can then think of High Level Strategy in the following terms:

1. When should one switch to a different Game State?
2. How should the roles be assigned among the robots?
3. What skill should a robot be invoking?

<sup>2</sup>Interestingly, soccer did not prove to be the best inspiration for successful robot soccer strategies. In many ways, the robotic game is much more like hockey and basketball than soccer.

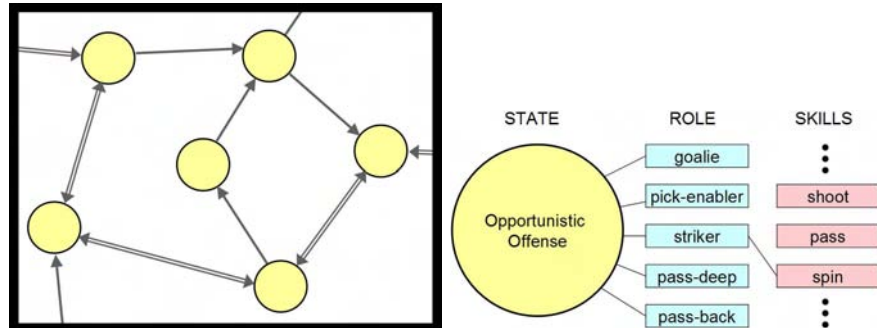


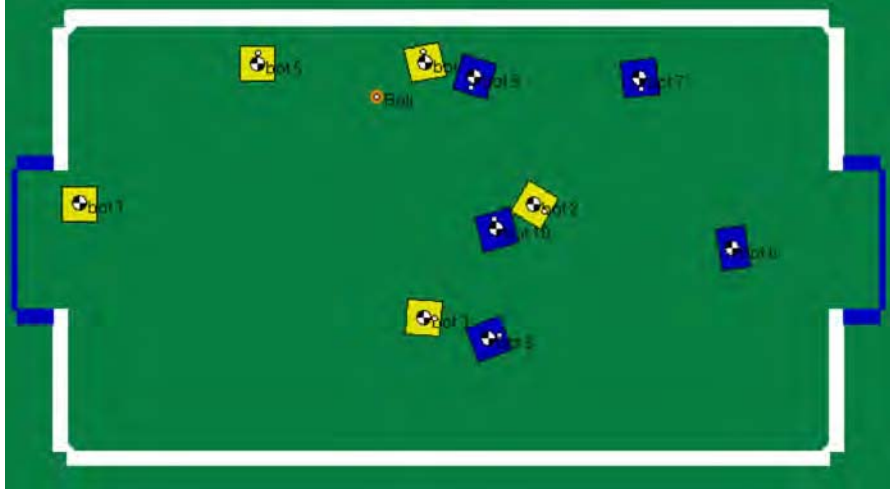
Fig. 8. LEFT: Directed graph with Game States. RIGHT: Assigning roles and skills.

The transitions between Game States were in general not deterministic. In particular, randomness was introduced, especially on offense, to allow the exploration, and exploitation, of an opponent’s weaknesses; reinforcement learning could then be used to alter the transitional probabilities based on various performance metrics, such as the position of the ball, scoring opportunities, etc. State overlap and hysteresis were used to prevent rapid transitions between Game States. An alternate approach, which was not pursued but is clearly of interest, is to design the system such that Game State transitions are “smooth” and thus do not require overlap and hysteresis. From a conceptual standpoint, this seems much more natural; in most situations, it does not make sense for the behavior of the system to change dramatically when a transition from one Game State to the next is made. For others, however, it certainly does make sense to have discontinuities; a free kick, for example.

Roles were generally assigned based on a cost function specific to the current Game State, which was in general not only a function of the current state of the system, but also the predicted (future) state of the system. A greedy algorithm, based on the relative importance of the roles, was found to yield performance similar to an exhaustive search. An interesting direction that was not pursued was to extract the relative ordering required for a greedy search from repeated runs of exhaustive searches.

The decision of which skill a robot should use was similarly based on a local cost function, which was in general a function of the future, predicted state of the system. This is, in fact, a simplified version of what was actually implemented. Skill selection could actually be coordinated among the robots. It was observed, however, that skill coordination was difficult to implement and only helped in limited circumstances.

High Level Strategy development was greatly aided through the use of a dedicated simulator, such as the one depicted in Fig. 9. The main use of the simulator was for designing the directed graph of Game States and for designing roles. Skill development, however, was typically performed directly with the robots. It was thus a design requirement that the robot skills satisfied the specifications used in the simulator, or conversely, that robot skills were faithfully captured by the simulator.



**Fig. 9.** Screen shot of first simulated RoboCup game, played in February 1999. The simulation engine was Working Model 2D, which was interfaced to MATLAB for real-time control.

## 5 Lessons Learned

The underlying system architecture is crucial to the success of a team. In particular, it should be modular and easy to adapt, and should allow a large number of individuals to concurrently develop the system. This modularity should appear at all system levels: from the high level decision makers down to the physical robots. While this may sound obvious, flexibility, adaptability, and modularity are often sacrificed for *potential* performance gains that are never realized.

Related to this point is the appropriate use of abstractions, again at all system levels. For example, the use of motion primitives insulates the High Level Strategy block from the details and complexities of robot motion control, which allows individuals with very little knowledge of robotics to design successful high level strategies.

A solid understanding of feedback, dynamics, and control is critical. For example, feedback is an excellent tool for making subsystems modular and thus easier to use as building blocks by non-experts. The appropriate use of filtering, estimation, and prediction can greatly simplify the interface between the robots and the high level algorithms that ultimately control them.

The final point is related to the objectives of the competition itself. Many teams use the RoboCup competition as a testbed for “artificial intelligence” and machine learning. In order to do well at the competition, however, it was much more useful to design a system that readily accepted inputs from humans and was readily amenable to redesign than to develop a system that “learned” on its own. One of the keys to our success in the RoboCup competition was our ability to very quickly encode human information into the system, such as implementing new strategies based on the scouting of opponents.

Not only was this important for winning the competition, it is also much more technologically relevant. There is nothing that currently comes close to the problem-solving abilities of humans, especially in environments where decisions must be made quickly. It thus makes sense to push research in a direction that utilizes these human assets, rather in one that marginalizes them. By doing so, we will gradually, but incessantly, reduce the amount of human input required to supervise and operate these multi-asset systems, and move that much closer to building truly “intelligent” systems.

## References

1. R. D’Andrea. Robot soccer: A platform for systems engineering. *Computers in Education Journal*, 10(1):57–61, 2000.
2. M. Asada, R. D’Andrea, A. Birk, H. Kitano, and M. Veloso. Robotics in edutainment. In *IEEE International Conference on Robotics and Automation*, pages 795–800, 2000.
3. R. D’Andrea and J. W. Lee. Small size league winner: Cornell Big Red. *AI Magazine*, 21(3):41–44, 2000.
4. P. Stone, M. Asada, T. Balch, R. D’Andrea, M. Fujita, B. Hengst, G. Kraetzschmar, P. Lima, N. Lau, H. Lund, D. Polani, P. Scerri, S. Tadokoro, T. Weigel, and G. Wyeth. RoboCup-2000: The fourth robotic soccer world championships. *AI Magazine*, 22(1):11–38, 2001.
5. P. Stone, T. Balch, and G. Kraetzschmar, editors. *RoboCup-00: Robot Soccer World Cup IV*. Lecture Notes in Computer Science. Springer, 2001.
6. R. D’Andrea, P. Ganguly, T. Nagy, and M. Babish. The Cornell robot soccer team. In P. Stone, T. Balch, and G. Kraetzschmar, editors, *RoboCup-00: Robot Soccer World Cup IV*, Lecture Notes in Computer Science. Springer, 2001.
7. T. K. Nagy, R. D’Andrea, and P. Ganguly. Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46:47–64, 2004.
8. O. J. M. Smith. Closer control of loops with dead time. *Chem. Eng. Progress*, 53(5):217–219, 1957.
9. E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.
10. M. Asada and H. Kitano, editors. *RoboCup-98: Robot Soccer World Cup II*. Lecture Notes in Computer Science. Springer, 1999.

