Big Red: The Cornell Small League Robot Soccer Team

Raffaello D'Andrea, Jin-Woo Lee, Andrew Hoffman, Aris Samad-Yahaya, Lars B. Cremean, and Thomas Karpati

Mech. & Aero. Engr., Cornell University, Ithaca, NY 14853, USA
{rd28, jl206, aeh5, as103, lbc4, tck4}@cornell.edu
http://www.mae.cornell.edu/robocup

Abstract. In this paper we describe Big Red, the Cornell University Robot Soccer team. The success of our team at the 1999 competition can be mainly attributed to three points: 1) An integrated design approach; students from mechanical engineering, electrical engineering, operations research, and computer science were involved in the project, and a rigorous and systematic design process was utilized. 2) A thorough understanding of the system dynamics, and ensuing control. 3) A high fidelity simulation environment that allowed us to quickly explore artificial intelligence and control strategies well in advance of working prototypes.

1 Introduction

In this paper we describe Big Red, the Cornell University Robot Soccer team. The success of our team at the 1999 competition can be mainly attributed to three points:

1. An integrated design approach; students from mechanical engineering, electrical engineering, operations research, and computer science were involved in the project, and a rigorous and systematic design process[6] was utilized.
2. A thorough understanding of the system dynamics, and ensuing control.
3. A high fidelity simulation environment that allowed us to quickly explore AI and control strategies well in advance of working prototypes.

The paper is organized as follows. In Section 3, we describe the electrical and mechanical aspects of the project, followed by a description of the global vision system in Section 4. The team skills are described in Section 5, followed by the artificial intelligence and strategy in Section 6. We include some of the other features of our team in Section 7.

2 Team Development

Team Leader: Raffaello D'Andrea[Assistant Professor]

M. Veloso, E. Pagello, and H. Kitano (Eds.): RoboCup-99, LNAI 1856, pp. 49–60, 2000.
Springer-Verlag Berlin Heidelberg 2000

Team Members:
Dr. Jin-Woo Lee[Visiting Lecturer]
Andrew Hoffman[Master of Engineering student]
Aris Samad-Yahaya[Master of Engineering student]
Lars B. Cremean[Undergraduate student]
Thomas Karpati[Master of Engineering student]
Affiliation: Cornell University, U.S.A
Web page http://www.mae.cornell.edu/RoboCup

3 Electro-mechanical System

3.1 Mechanical Design

The Cornell University team consists of two mechanical designs, one for the field players and the second for goalkeeper. All of the robots have a unidirectional kicking mechanism powered by one solenoid (two for the goalkeeper).

The robots have a design mass of 1.5 kg, a maximum linear acceleration of 5.1 m/s^2, and a maximum linear velocity of 2.5m/s. The goalkeeper has a different design from the field players. It is equipped with a holding and kicking mechanism that can catch a front shot on goal, hold it for an indefinite amount of time, and make a pass. All of the designs were performed using ProE[10].

Listed below are the main characteristics of our robots:

Characteristic	Goal Keeper	Field Player
Weight	1.78 kg	1.65 kg
Max. Acceleration	5.90 m/s^2	5.10 m/s^2
Max. Velocity	1.68 m/s	2.53 m/s
Max. Kicking Speed	4.18m/s	2.6 m/s
Operating time	30 min per battery pack	
Special function	Ball Holding mechanism	

3.2 On-Board Electronics

The main function of the on-board electronics is to receive left and right wheel velocity signals via wireless communication and to implement local feedback control to ensure that the wheel velocity were regulated about the desired values. Considering the speed, memory space, I/O capability, and the extension flexibility, 16bit 50MHz microcontrollers are used.

In order to get a precise kick, a ball detecting system separate from the global vision system is implemented. An infrared system is used to detect the ball. It informs the microcontroller when the ball has come into contact with the front of the robot. When the global vision system makes the observation that the robot

is in a position to kick, a command is sent to inform the robot to kick the ball if and only if the infrared system detects the ball. All the capture and layout for the on-board electronics were performed in-house using OrCad[11].

3.3 Communication

After careful considerations and trade-off analysis, the wireless communication was limited to one-way transmission from the global AI workstation to each robot. The main justification for the decision is the lack of on-board local sensing information. The one-way transmission saves the communication time, as compared to the two-way communication, and simplifies the AI strategy and the on-board firmware. The wireless communication system takes 12.5ms to transmit the information from the AI workstation to the robots.

With the experimentally verified assumption that the robots do not drift far from the desired position between frames, the need for local sensing and correction is minimal. Based on the current design, the robots can drift 5cm at maximum speed. For debugging purposes, each robot has the capability to transmit data back to the AI workstation.

4 Visual Tracking Algorithm

A dedicated global vision system identifies the ball and robot locations as well as the orientation of our robots. In order to determine the identity of each robot and their orientation, blob analysis[5] is used as a basic algorithm. The vision system perceives the current state of the game and communicates this state to the AI workstation allowing decisions to be made in real-time in response to the current game play. The end result of the vision system is the reliable real-time perception of the position and velocity of the ball and the players, and also the orientation and the identity of the Cornell players. The vision system captures frames at a resolution of 320x240 and a rate of 35 Hz.

4.1 Interest Determination

Color segmentation of a frame often results in spurious blobs that do not correspond to the ball, or the robots. These points can be resultant from areas outside of the field, highlights from the lighting, deep shadows, the goals, and aliasing. Computation time of features of these blobs can result in a significant slowdown in system performance. To eliminate this slowdown and to produce a clean image of only the objects of interest, a single frame is captured and saved previous to the beginning of game play. This frame consists of the empty field only, without robots or the ball. This frame is later subtracted from the currently captured frame producing a difference image. Regions of this image where the disparity is high are postulated as areas where objects of interest lie.

4.2 Color Segmentation and Feature Extraction

Once areas of interest are determined, color segmentation is performed on the image. The segmentation is done by independently thresholding each of the Red, Green, and Blue color channels and performing a logical AND on the results of color thresholding. This logical operation extracts a sub-cube from the RGB color space. All objects are then classified based on the sub-cube that the corresponding blob falls into. This approach is well suited for the colors that are determined by the RoboCup Federation. The ideal pure colors that are defined: Orange (which is mainly Red), Green, Blue, Yellow, White, and Black, can be found on the corners of the RGB color cube. Two remaining colors exist and are used for our purposes, Cyan for initial orientation information and Magenta for robot identification. Since these eight colors are so separated in RGB color space, no color space conversion is performed on the input image, which is computationally costly.

Once the color segmentation is completed, blob features are computed including position, size, and perimeter length. These features are then used to filter any salt and pepper noise that may have been the result of incorrect color thresholding.

4.3 Tracking

Further rejection of false object classification is performed in the tracking stage. During tracking each orientation and identification blob is attempted to be registered to the blobs that correspond to the appropriate Cornell team color. All blobs that are not located within an appropriate physically realizable distance from the team color blob are thrown away. This step will reject any colors that are of interest, but are found on the opponent robots, for example. The team marker blobs which have an orientation marker and one or more identification markers registered with them are considered initially for identification and localization. The other team markers are afterward considered if there are any robots that are not found in the first set.

From these markers, and initial orientation is computed and the positions of the identification marker with respect to this orientation marker. The identification markers can be located in three of the four corners of the robot cover. The cover is divided into four quadrants, and each marker is classified by the angle that is produced relative to the orientation marker. The pattern of the identification markers can then be determined and the robot identified invariant to the robot orientation on the field. Once the robot pattern is identified, the final orientation is computed using all of markers on the cover.

Robot identification is also simplified by the physical constraints imposed by the maximum velocity and acceleration attainable. Each candidate position for a specific robot is compared to a predicted location and velocity determined by the two previous locations. If the position is outside the physically realizable radius of this predicted position, the candidate is rejected. Among several candidate positions that are located within this attainable radius, the necessary velocity

to reach that position is computed and the position that most closely matches the predicted position and velocity is chosen as the true position of the robot.

4.4 Filtering

Although, the resolution of the camera is quite high and the blob analysis can compute the center of gravity of a blob to sub-pixel accuracy, these centers of gravity contain noise that may be modeled as white Gaussian noise in the system. While position calculations are fairly accurate, the orientation calculation suffers from this noise. The spatial proximity of the orientation marker and the team marker on the cover of the robot results in orientation errors of up to 10 degrees. To compensate for these errors all of the markers on top of the robot are used for the orientation. The true positions of the markers are known, thus the optimal rotation of the robot can be computed by using a least squares fit of the perceived marker locations and the actual locations on the cover. This fit becomes more precise as the number of markers on the cover of the robot increases.

5 Skills

The sophistication of the trajectory control algorithms described below together with very tight PID velocity control[4] enable our robots to get to a desired final state (of position, orientation and wheel velocities) in a fast manner.

This, combined with a prediction algorithm[3] for the ball, makes for effective real-time interception for both a stopped ball and a moving ball. The limitation on robot speed of maneuverability comes primarily from a system latency, described later in this paper.

Ball control is achieved with a front surface that is slightly recessed from the front corner bumpers, nominally allowing a player to change the direction of the ball's motion. An energy absorbing contact surface affords greater control. Dribbling is not a dominant skill. Passing is accomplished in an emergent manner, as a result of clever positioning of players that are not assigned to the ball.

Kicking is accomplished by a unidirectional solenoid with a front plate attached to its shaft. Robots will only kick in potential goal-scoring situations, and the timing of the kick is done with the use of an infrared sensor circuit that detects when the ball is directly in front of the robot. Typical kicks impart an additional 1 m/s to the ball.

The goalkeeper design is independent of the field player design, and thus the goalkeeper exhibits significantly different skills. The goalkeeper is equipped with a holding and kicking mechanism that can catch a front shot on goal, hold it for an indefinite amount of time required to find a clear pass to a teammate, and make this pass.

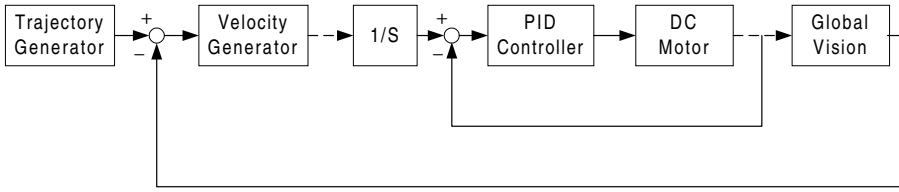

Fig. 1. Schematic diagram for Robot Control

6 Artificial Intelligence

6.1 Role Based Strategy

The artificial intelligence subsystem is divided into two parts. The high-level AI takes the current game state (robot and ball positions, velocities, and game history) as input, and determines the role of each robots, such as shooter, defender, mid-fielder, goalie, etcetera. Please see [2], and the references therein, for a thorough description of the application of role based systems in robotic soccer. Once the role is determined, desired final state such as time-to-target, robot orientation and robot velocity at the final position are computed from the current state. More than 20 roles are preprogrammed. The low-level AI subsystem resides on the each roles, and generates the trajectory to the target point and computes the wheel velocities to transmit to a robot.

6.2 Trajectory Control

The task of low-level AI is to generate trajectories and to control the robot to follow the trajectories. It takes as inputs the current state of the robot and the desired ending state. The current state of a robot consists of the robot x and y coordinates, orientation, and left and right wheel velocities. A desired target state consists of the final x and y coordinates, final orientation, final velocity as well as the desired amount of time for the robot to reach the destination.

Compared to reactive control strategies, such as those in [1] for example, we perform a global trajectory optimization for each robot and take advantage of the mechanical characteristics of the robots. Two position feedback loops are employed for the robot's trajectory control. The first is a local feedback loop and the other a global feedback loop. The local feedback loop resides on the microcontroller of each robot and is in charge of controlling the motor position[4]. The global feedback control also has a position feedback loop via the global vision system and makes the robot follow the desired trajectory. These two position feedback controls improve the robot's staying performance. The performance enhancement shows up especially when the goalie is facing an opponent robot. The desired velocity of each of the robot wheels are generated and then transmitted

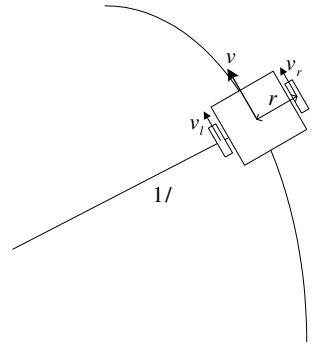

Fig. 2. Trajectory Generation

to the robots through the RF communication link at every sixtieth of a second. Fig. 1 shows the schematic diagram for the entire trajectory control loop.

The low-level AI needs to be efficient and robust to imperfections in robot movement. Currently, our algorithm can run more than 60 times per one computation cycle, which is sufficient considering it only needs to be run at about four times per cycle (for four robots excluding the goalkeeper).

This complex problem is solved by breaking the problem of trajectory generation into two parts. The first part generates a geometric path. The second part calculates wheel velocities such that the robot stays on the path and completes it in the allocated time.

Generating a Geometric Path Our geometric path is represented by two polynomials in the x and y coordinates of the robots. The x coordinate polynomial is a fourth-degree polynomial and the y coordinate polynomial is third degree.

$$x(p) = \sum_{k=0}^{4} \alpha_k p^k \tag{1}$$

$$y(p) = \sum_{k=0}^{3} \beta_k p^k \tag{2}$$

The task is to solve for the nine polynomial coefficients for a particular path requirement[7]. The 9 constraints on the polynomial path are: initial x coordinate, initial y coordinate, initial orientation, initial curvature (determined by the initial left and right wheel velocities), initial speed, final x coordinate, final y coordinate, final orientation, and final speed.

Generating Wheel Velocities Every point on the geometric curve has a curvature value, which defines a relationship between the left wheel velocity v_l

and the right wheel velocity v_r at that point in the curve. This relationship is:

$$v = (v_l + v_r)/2 \tag{3}$$

$$v_l(1 + \kappa \cdot r) = v_r(1 - \kappa \cdot r) \tag{4}$$

where κ is the curvature of the path, and r is the half distance between the two wheels and v is the forward moving velocity of the robot(See Figure 2). Thus, we simply need to choose a forward moving velocity of the robot to solve for v_l and v_r at every point on the curve, which can then be sampled at the cycle rate of our AI system. Obviously, the forward moving velocity is constrained by the time-to-target as well as mechanical limits of the robot.

Even though each run of this algorithm generates a preplanned path from beginning to end, it can be used to generate a new path after every few cycles to compensate for robot drift. The continuity of the paths generated is verified through testing. However, this algorithm breaks down when the robot is very near the target because the polynomial path generated might have severe curvature changes. In this case, the polynomials are artificially created (and not subject to the above constraints) on a case-by-case basis, and these are guaranteed to be smooth.

7 Other Team Features

7.1 Vision Calibration

The vision calibration consists of 4 main parts. They are:

- barrel distortion correction
- scaling
- rotation
- parallax correction

The barrel distortion correction is performed using a look-up table to map a point in image-coordinates into a new coordinate equidistant coordinate system. Due to the necessity for a wide angle lens barrel distortion becomes a significant problem in the image processing. Barrel distortion is a function of the lens of the camera and is radially symmetric from the center of the image. To invert the distortion, points are measured from the center of the image to the corner of the image. Since the points are measured equidistantly, a scaling can be done to convert corresponding points in image coordinates, which tend not to be equidistant. A look-up table is generated which contains the scaling factor and indexed by the distance from the center point in the image. Points between the indices are linearly interpolated from the two surrounding points. This transformation ensures that straight lines in reality are mapped back into straight lines.

The scaling is then computed such that the sides of the field are computed as accurately as possible. These values are computed so that the transformation from image coordinates to field coordinates is accurate for the center of each of

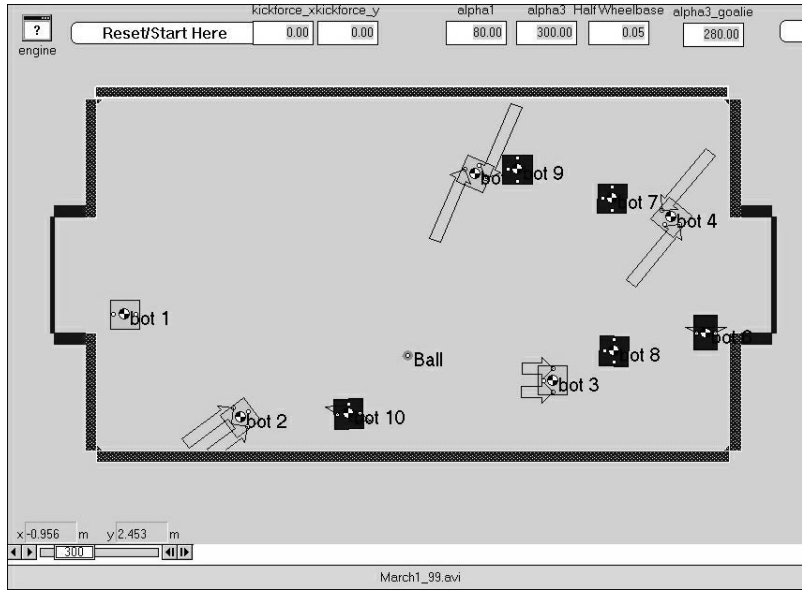

Fig. 3. Testing platform for the Simulation

.

the sides of the field, i.e. the x-coordinate of the center of the goals, and the y-coordinate for the centers of the lengthwise walls. This anchors the transformation to the sides of the field, ensuring that the walls of the field correspond to exactly where the artificial intelligence system expects them to be.

Since the camera cannot be mounted perfectly, the rotations about the center axis of the camera also need to be taken out. The points are rotated so that the sides of the field have constant x-coordinates along the widthwise walls and constant y-coordinates along the lengthwise walls.

Finally the parallax error that results from differences in object height is removed by scaling the x- and y-coordinates proportionally to the distance that the object is from the center of the camera projected onto the field plane.

7.2 High Fidelity Simulation

To provide a realistic testing platform for our artificial intelligence system, we have constructed a simulation of the playing field that models the dynamics of our environment.

The dynamic modeling of our system is performed by a Working Model 2D[8] rendering of the complete playing field. The model includes two teams of five individual players, the game ball, and the playing field. Real world forces and constraints are modeled, including the modeling of the motion of the tires and the inertia of the robots and ball. Additionally, the physical interactions between

the players and each other, the ball, and the playing environment are all modeled in Working Model's two dimensional environment.

The simulator accepts external input and output 60 times per simulated second, the rate at which the artificial intelligence operates, and the rate at which new robot commands are issued. To simulate the time lag and noise we encounter in our real world simulation, the Working Model parameters are passed into Matlab[9], where random noise, error, and delay are introduced to model the limitations of our vision and communication systems. In addition, the kicking mechanisms and simple referee functions are managed, such as the detection of a team goal, and the reset of the playing field. This information is then passed to the artificial intelligence module. Information normally transmitted across the communications link is then passed back to Matlab from the artificial intelligence module, and is interpreted in Matlab before it is applied to the model of our system, to simulate the delay associated with our real world communications link.

The transition of our intelligence code between the simulated environment to the real playing environment is fairly smooth. However, there are some underlying differences that need to be considered when making the transition. In the simulator, the intelligence must run synchronously with the rest of the system, unlike in our final playing environment. In addition, simpler feedback control mechanisms are implemented into the physical model to increase the speed of the simulator, which are slightly different than the control methods that we use on our actual robots. Despite these limitations, we have found the differences between the environments to be small enough that we can make the transition between them without the need to significantly alter the functionality of the system intelligence and control code.

In the time before we had a fully operational real world system, the simulator provided us with a means of testing the artificial intelligence play-by-play, allowing code and strategy development to begin a full four months before the first robots were built. In addition, it allowed us to run competing strategies and full games without the need for a full complement of ten robots and two workstations running different intelligence algorithms simultaneously.

The simulator is a simple and manageable platform that allows us to recreate the real-world problems that exist in our system. At the same time, it removes several annoying physical constraints of the real-world system, such as limited battery life and operating time, providing a more convenient environment for new algorithms to be tested. Because the simulator performs more reliably than the real-world system, problems can be traced more quickly and more reliably to problems within the intelligence code. The simulator is a convenient and fairly accurate rendering of our real-world system, and an invaluable tool in the design and implementation of our artificial intelligence system.

7.3 System Delay Test

In any feedback control system, a change in the system delay can cause unwanted oscillations and loss of system control. To prevent unwanted oscillations, we have

created a simple testing procedure to accurately measure our system delay. We measure the delay by sending a command to a robot to change the state of the playing field, and then we measure the time needed to detect the arrival of the change back at the place in the system where we issued the command.

There are two major reasons that we chose to implement the delay measurement in this manner. First, this method of measurement provided us with an accurate representation of the system delay that could be used when modifying the trajectory controller, which is responsible for modeling the system and properly handling the system feedback. Second, a quick measurement of the system delay allows us to easily check if the system is functioning normally.

The results of our delay measurements have reflected a total system delay of approximately 83 to 117 milliseconds. The total delay time can be attributed to specific components within the system. Our intelligence contributes a single frame delay of about 16 milliseconds (ms) to the total system delay. The communications link approximately adds an average of 13.3 ms of delay due to the time needed to buffer our 13 byte packets into the device, send the packets across the data channel, and to decode the packets at the robot.

The time that is needed for the vision system to capture the field state and relay it to the intelligence system is approximately 45 ms. This can be attributed to a delay of 16 ms to capture a new frame from the camera and the time needed to interpret a single frame, which is approximately 29 ms.

The vision information is incorporated into the next artificial intelligence cycle, which begins a new cycle every 17 milliseconds. The entire system delay breakdown gives us a minimum system delay rate of 75 ms, with the possibility of additional delay due to the asynchronous nature of the links between the camera, vision, and artificial intelligence subsystems.

8 Conclusion

Even though our team performed well at the competition last year, there are many subsystems and components that need to be improved. The main ones are outlined below:

- A more robust vision system. The current vision system performs well when operational, but does fail on occasion. In addition, it takes a very long time to calibrate the system. One of our objectives for next year is to construct a reliable vision system that can be set up in less than 30 minutes.
- Role coordination. This will allow us to implement set plays.
- More refined trajectory generation, obstacle avoidance, and trajectory control.
- Reduce the system latency.
- Innovative electro-mechanical designs.

Acknowledgment

The authors would like to thank all of the members in Cornell RoboCup team for their help in the design and construction of the system, Professors Bart Sel-

man and Norman Tien for helpful advice and comments, and Professor Manuela Veloso for her insights and information on the CMU AI during a RoboCup lecture at Cornell University.

References

1. RoboCup-98:Robot Soccer WorldCup II, Minoru Asada and Hiroaki Kitano(Ed.), Springer Verlag. 1998
2. Manuela Veloso, Michael Bowling, Sorin Achim, Kwun Han and Peter Stone. The CMUnited-98 Champion Small Robot Team. In RoboCup-98:Robot Soccer WorldCup II, Minoru Asada and Hiroaki Kitano(Ed.), Springer Verlag. 1999
3. Eli Brookner. Tracking and Kalman Filtering Made Easy. A Wiley-Interscience Publication. 1998
4. Joseph L. Jones and Anita M. Flynn. Mobile robots: Inspiration to Implementation. Wesley. 1993
5. Rafael C. Gonzalez and Richard E. Woods, Digital image processing, Addison-Wesley, 1992
6. S. Blanchard and W.J. Fabrycky. System Engineering and Analysis. Prentice Hall, 3rd Edition, (1997)
7. G. Thomas and Ross Finney. Calculus and Analytic Geometric. Addison Wesley. 1996
8. Working Model. Knowledge Revolution, San Mateo, CA.
9. Matlab, The Mathworks Inc., Natick, MA.
10. Pro/ENGINEER. Parametric Technology Corporation, Wahlthan, MA
11. OrCAD. OrCAD Inc., Beaverton, OR.