

DESIGN AND DEVELOPMENT OF AN AUTONOMOUS SOCCER-PLAYING ROBOT

Steven A. R. Olson

**Department of Electrical and Computer Engineering
Brigham Young University**

Chad S. Dawson

**Department of Electrical and Computer Engineering
Brigham Young University**

Jared Jacobson

**Department of Electrical and Computer Engineering
Brigham Young University**

Faculty Advisors:

James K. Archibald

Randal W. Beard

ABSTRACT

This paper describes the construction of an autonomous soccer playing robot as part of a senior design project at Brigham Young University. Each participating team designed and built a robot to compete in an annual tournament. To accomplish this, each team had access to images received from a camera placed above a soccer field. The creation of image processing and artificial intelligence software were required to allow the robot to perform against other robots in a one-on-one competition. Each participating team was given resources to accomplish this project. This paper contains a summary of the experiences gained by team members and also a description of the key components created for the robot named Prometheus to compete and win the annual tournament.

KEY WORDS

Robot Soccer, Senior Design Project, Artificial Intelligence, Autonomous Control, Histogram, Normalized Color, Robot Vision.

INTRODUCTION

The 2000-2001 robot soccer tournament at Brigham Young University was the culmination of three years of collegiate undergraduate experience for the twenty-seven students who participated in this multi-university event. This Brigham Young University sponsored event was created by Dr. James K. Archibald and Dr. Randal W. Beard for students as a senior design project, similar to other universities which use robots as part of a design project (see [4] and [5]).

The goal of robot soccer as a senior design project was to provide a team-based multi-disciplinary approach to a real world problem, gaining experience by trying to meet hard deadlines and working with others to provide results that no individual team member could create alone. In addition, other skills were developed through monthly presentations to faculty and the writing of documents meant to chronicle everything from team thought processes to the desired specifications of the finished robot.

Using these processes and by creating goals for our team, we created our robot, Prometheus [1], to compete in the 2000-2001 BYU robot soccer tournament.

COMPETITION FORMAT

Competition Rules

Each round of the robot soccer competition consisted of six thirty-second, one-on-one shootouts (called “matches”), with each robot starting as the offensive player for three of the matches. The robots and ball were required to be placed on the field according to certain time constraints which will not be repeated here for the sake of brevity[2]. Both the offensive and defensive robots could score, and the winner of the round was the robot that scored the most goals in the six matches. When play began for a given match the robots’ operators directed the robot to start, and from then until the end of the match the robot needed to play completely autonomously.

Robots were not allowed to move the ball in such a way that the opponent robot could not take it away. “At least one half the ball's diameter must be visible at all times from a view directly overhead the side of the robot in contact with the ball” (Archibald 2000). The robot was allowed to have incidental contact with its opponent, but aggressive or uncontrolled behavior was penalized.

Field Layout and Provided Hardware

The field was nine feet by five feet with the standard soccer markings, as shown in **Figure 1**. Vision was provided to the robots via a camera suspended over the field, which dumped data into a computer workstation through a frame-grabber at thirty frames per second.

Communication between the workstation and the robot was provided by RF transmission using a

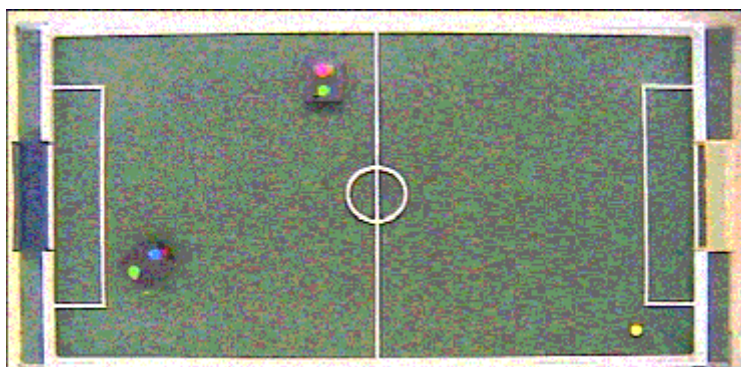


Figure 1. Field overhead view.

Linx¹ HP Series-II transmitter and a corresponding receiver operating between 903.37 and 921.37 MHz. Commands received from the workstation were processed using a Handy Board, which is a small microcontroller board designed by MIT.

Robot Design Rules

Offensive and defensive players were identified and distinguished by a pair of colored patches atop the robot. Each robot had one green patch that served as a help for orientation. The other patch was red for the offensive robot and blue for defensive robot. These colors were switched on the robots between each match. All other visible parts of the robot were required to be black or dark gray. Each robot was required to fit completely within a seven-inch cube, including its body, sensors, patches, kickers, etc., with the only exception being its antennae, which could extend beyond the limits of the cube.

DESIGN

HARDWARE

The drive system consisted of two motors mounted on the middle of an aluminum chassis (see **Figure 2**). The motors were arranged in a differential drive configuration such that each had complete freedom from the other. Both motors could be turned on in the same direction for forward movement or in opposing directions to spin the robot. The motors were driven by a pulse width modulated signal originating from the MIT Handy Board such that each motor could move at any speed, enabling turns to be made with a radius anywhere from zero to infinity.

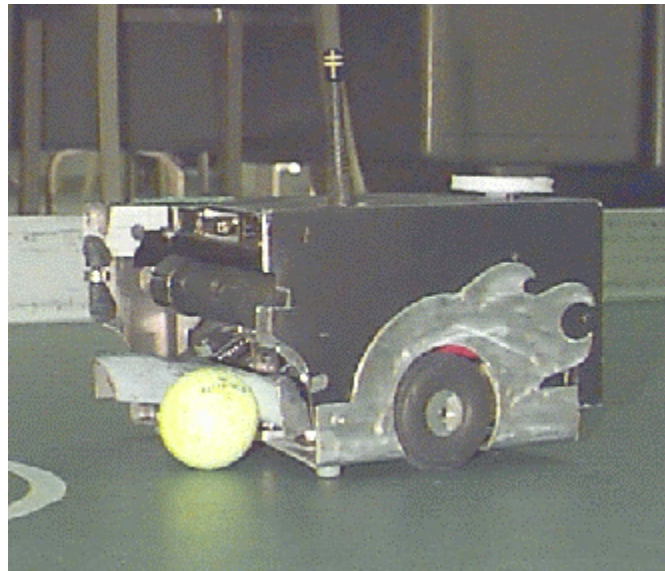


Figure 2. Prometheus.

The Handy Board sent these PWM signals to the motors through two of four motor drivers it carries on board. One of the remaining drivers was used to activate a relay which shorted the contacts of the motors together. With the contacts shorted together the back EMF of the motors resisted any movement whatsoever. This allowed the implementation of brakes on the robot, which proved to be very effective. The remaining driver was used to power an electrically activated pressure valve. This valve controlled the flow of CO₂ to a pneumatic kicker which was installed on the front of the robot.

When the valve opened, the kicker would activate and when closed the kicker would

¹Linx is a registered trademark of Linx Technologies, Inc.

automatically retract to its position inside the robot. The kicker had about 1 ½ inches of travel and the high pressure CO₂ cartridge provided ample power to the kick.

Of course, to successfully kick the ball, it had to be in the right place. A plow was needed on the front of the robot that would keep the ball in range of the kicker. Our design constraints were that ½ of the ball had to be visible from both the top and side of the robot at all times. In addition, teams were not allowed to “grab onto” the ball such that it could not be stolen by an opponent. A V-shaped design, (shown in **Figure 3**), was settled on for the plow in order to funnel the ball toward the center of the robot.

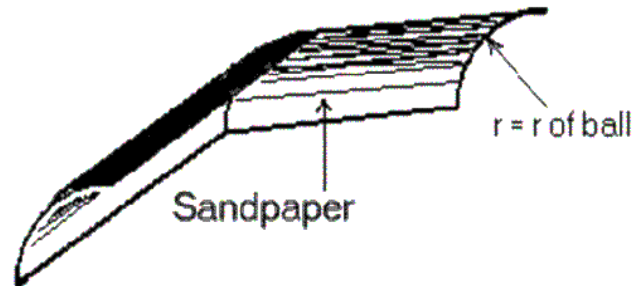


Figure 3. Plow Design

In addition, the plow was curved to have the same inside radius as the radius of the ball.

The inside surface of the plow was covered with sandpaper. The plow was mounted on springs in order to put downward pressure on the ball. The ball then would get “wedged” in between the plow and the playing field. The robot could push the ball around like a building block because it did not roll when the robot was moving forward. However, another robot could still steal the ball from our robot, granted, with more than an inconsequential effort. The robot could also perform larger radius turns while maintaining control of the ball. In addition, the plow only contacted the ball over the top half. This allowed the kicker to move freely underneath the plow.

Again, all of the electronics on board were controlled by the Handy Board. The Handy Board had an RS-232 input which was used to receive the commands from a radio receiver which received commands from a transmitter hooked to the PC. The commands consisted of four bytes of data which were decoded to produce speeds for the motors. Some special commands were also used to signal the braking routine and the kicking function.

ARTIFICIAL INTELLIGENCE

Most of the soccer-playing robots were very similar in appearance. The key feature that differentiated how the robots performed during competition was the control software that dictated how they operated. Data transmitted serially to Prometheus from the controlling computer could only communicate basic information on how much voltage to apply to the robot’s motor drivers. The artificial intelligence therefore was required not only to decide where to place the robot, but also the precise detail in what voltages needed to be sent in order to perform the desired action. To accomplish these tasks, Prometheus’ artificial intelligence was broken up into three fully integrated components: high-level artificial intelligence, offensive and defensive functions, and robot control primitives.

The high-level artificial intelligence was organized in a branch structure. Through the use of a graphical user interface, strategies before each match were selected—much like a coach calls plays during a game. These initial inputs were given to the artificial intelligence system which used them as a starting point on one of its branches. After each match began, no more input was

allowed to be given by the user and the robot was required to use the information supplied by the vision system. At this point, the high-level artificial intelligence would begin to process this information and determine which additional branch to take. Each choice was dependent upon the locations of the robots, the past actions of the robot, and the current and past locations of the ball.

The offensive and defensive functions were created to be abstractions of the lowest level functions. The functions in this group included complex tasks that required the use of many of the control primitives. Some of the characteristics of the functions in this group were miniature branch decision structures and reliance upon proximity checking.

One of the biggest problems with decision-making for the robot that this class of functions needed to overcome was the issue of accuracy. For example, Prometheus' ricochet function, (illustrated in **Figure 4**), needed to capture the ball, determine where to turn to in order to fire the ball into the wall using its pneumatic kicker, and then turn to that angle and fire. The issue of accuracy arose in turning to the precise angle needed to accomplish this task. This was resolved through the use of "dead zones". Dead zones were areas which allowed for the hardware imprecision by determining that if an angle was within certain bounds of the desired angle, or if the robot was within a certain radius of a desired position, the functions would consider the robot close enough to the desired angle or location to continue.

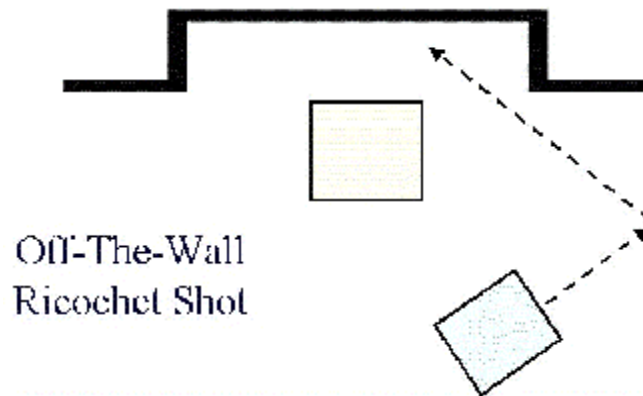


Figure 4. Ricochet Shot.

The robot control primitives were what really differentiated a mediocre robot from an excellent robot. These functions were the building blocks of everything else that was done on the artificial intelligence. The functions that are considered primitives included a function for position, one for turning to the correct angle, one for getting behind the ball, and a function to get Prometheus "unstuck" from the wall. Primitives used past and current information on velocities and position to determine the actual velocities that were sent to the robot. Precise locations were arrived at in the position function through the use of proportional integral derivative control which used the information of the changes in angle and velocity made in previous movements as well as the currently calculated velocities to create a set of velocities sent to the motors.

VISION

Two vision systems were written for the robot. They were used at different stages of the competition. One followed a system suggested by the faculty advisors in late November, 2000. The other was an adaptation of a vision system used by the University of Queensland RoboRoos team of 2000, that competed in the RoboCup robot-soccer tournament [3]. Both systems had strengths and weaknesses. The first will be called the normalized vision system (for reasons that will be described later) and the second will be called the histogramming vision system.

The normalized vision system normalized colors on the field to identify the color patches on the top of the robot. The same “color” of red could have a number of different intensities. That is to say, the proportions of red, green, and blue in the pixels were the same, but the values themselves were different. One pixel might have had a red, green, blue (RGB) value of (80, 20, 12), and another might have had an RGB value of (120, 30,18)—the colors were the same, proportionately, but the second pixel was brighter than the first.

To get the computer to recognize two reds of differing intensity as the same color, one could “normalize” the color: that is, one could reduce or increase the intensities of the pixels until they appear identical. If the red, green, and blue values of the pixel were viewed as orthogonal vectors, this would be analogous to normalizing the vector. Just as two vectors could point in the same direction but have differing length, the two pixels were the same color, but of differing intensities. If the two vectors were normalized, the colors became identical. Before normalizing the colors in the 24-bit-per-pixel color that came from the camera (256 values each of red, green, and blue intensity), there were a total of 256^3 distinct value combinations. After normalizing the colors (i.e. removing intensity information) the computer only had to deal with 256 distinct colors.

For example, if one had a pixel with red, green, and blue values r , g , and b , respectively, one could normalize it by resetting the values as follows:

$$\begin{aligned} r_{\text{norm}} &= 255 * (\text{float})r/(\text{float})(r + g + b) \\ g_{\text{norm}} &= 255 * (\text{float})g/(\text{float})(r + g + b) \\ b_{\text{norm}} &= 255 * (\text{float})b/(\text{float})(r + g + b) \end{aligned}$$

This normalizes the colors so that the sum of the red, green, and blue values of the pixel sum to 255. One can then check to see if the pixel is within the color bounds defined as “red.”

The histogramming vision system used a color histogram to reject the most common colors on the field under the assumption that the most common colors in the field image were the field colors. Three arrays of 256 color values were kept—one for red, one for green, and one for blue. For each pixel on the field the array value in the color array that corresponds to the color intensity of the pixel was incremented. For example, if a pixel with RGB value (20,156,83) was being added to the histogram, the red array at position 20 was incremented, the green array at

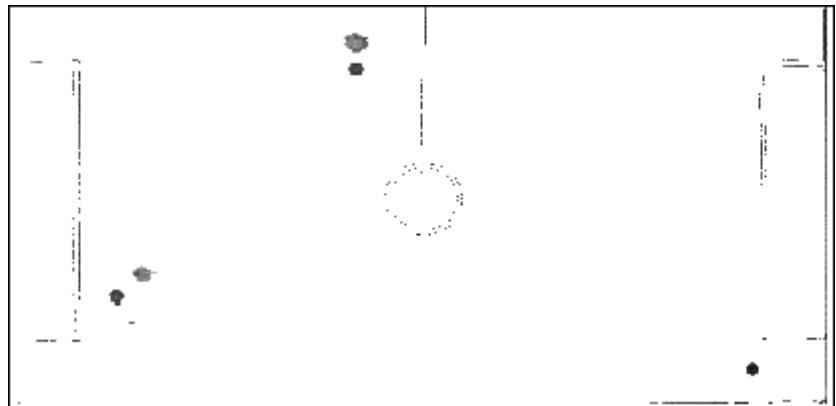


Figure 5. The field after histogram background rejection.

position 156 was incremented, and the blue array at position 83 was incremented. The algorithm would then keep track of the position with the most “hits” in the array. When the color histogram operation was complete, the upper and lower bounds of the region to be ignored were

widened until the values fell off below a certain level (see **Figure 5**). Patches on the field were located by performing a second pass through the field image using the color bounds defined by the histogram.

The histogramming vision system worked beautifully when lighting conditions were fairly uniform. That is, it worked nearly perfectly—at thirty frames per second—in lighting situations where the light was even across the field. For the practice competition and the preliminary (seeding) competition the field was in a room with uniform, fluorescent lighting, so the histogramming vision system was used. In the final robot soccer competition, however, the field had spotlights for lighting, which threw off the histogram. Consequently, the normalized vision system was used.

RESULTS AND CONCLUSIONS

This project was a wonderful introduction into the field of robotics. Throughout the eight months that our team spent working on Prometheus, we learned the nature of team dynamics and how teams can effectively accomplish goals. This senior design project also taught us of the importance of planning, scheduling, and using a top-down design methodology. Using these techniques, Prometheus was victorious in the eight-team field at the final tournament.

One of the largest factors in Prometheus' victory was its software. As we designed Prometheus, we discovered that the majority of the time spent on the system would need to be on the software. For this reason, we designed Prometheus' hardware early in the project and dedicated the remainder of the time with the software.

Our three-man team used their expertise to enhance the project. Each team member was given control over one facet of the project: hardware, vision, and artificial intelligence. The members then had their own design responsibilities and tasks to perform. As strategy meetings were held, each member was able to report on the status of their end of the project. In addition, team dynamics were quickly discovered and the team was organized with a project leader to keep the effort on schedule.

The project brought out the best in many teams besides our own. Several innovative hardware and software ideas were implemented. These included teams with rotating chassis, rollers to put a spin on the ball to better hold it, and several different kickers. Although only one team could win the tournament, all of the students who participated in this event greatly enjoyed their experience.

As this event is continued, students will continue to work on the foundation laid by those who completed this design project in previous years. New techniques for increasing the speed of the vision system, different approaches to artificial intelligence and more innovative hardware ideas are in store for the future. As this project garners more support, many more students will be able to enjoy this experience-building engineering competition.

REFERENCES

- [1] Team Prometheus Home Page (BYU Electrical and Computer Engineering)
<http://www.ee.byu.edu/ee/srprojects/robot/2000/prometheus/index.html>
- [2] BYU Robot Soccer Robot Regulations (BYU Electrical and Computer Engineering)
<http://www.ee.byu.edu/ee/srprojects/robot/>
- [3] Robocup 2000 <http://www.robocup2000.org/>
- [4] Crisman, Jill D., "System Design Via Small Mobile Robots," IEEE Transactions on Education, Vol. 39, No. 2, IEEE, Piscataway, NJ, May 1996, pp. 275-280.
- [5] Kruempelstaedter, Kerry, "Robocar: Unmanned Ground Robotics," Linux Journal, Issue 41, Seattle, WA, September 1997, pp. 16-22.