Effective Ball Handling and Control in Robot Soccer

Walter H. Johnson* Rob Franklin†
Department of Electrical and Computer Engineering
Brigham Young University
Provo, UT, 84602

Faculty Advisors
Prof. Randal W. Beard
Prof. James K. Archibald

Abstract

An autonomous soccer-playing robot was designed and constructed as part of the electrical engineering senior project at Brigham Young University. Unique physical features enable ball handling abilities. A front-side rotating drum retains the ball by applying backspin. A pneumatic kicker allows for fast accurate kicks. Robot movement control is performed by three different systems. The low-level velocity controller ensures output of desired forward and angular velocities. The position controller is used to match the robot's position with a desired position. Path generation directs the robot along a desired path at a specified velocity. The locations of the robots and ball on the soccer field is determined by the vision system. This document is a formal description of the unique designs and construction of the our team's robot.

1 Introduction

Since August of 1997, the Electrical and Computer Engineering (ECEn) department of Brigham Young University (BYU) has provided Robot Soccer as an option to fulfill the senior project requirement [1]. Robot Soccer is relates to the organizations RoboCup [2] and FIRA [3] in that soccer is the primary domain and AI and intelligent robotics research is fostered [4]. The senior project is intended to be a culmination of an undergraduate student's engineering study. Through the project, students gain an understanding of engineering methods and experience as they relate to the real-world. Each year, students' robots improve, because students and professors build from the experience of previous years.

The Robot Soccer regulations state that the robot must fit inside a cube with sides of a length of seven inches. Our robot was built to fit within the regulation cube. The field that the robots play on is 108 inches by 60 inches. A yellow golf ball is used as the soccer ball [5] [6]. The name of our robot, which is shown in Figure 1, was MaXimus.

This paper discusses the design and concerns as they relate to implementation of MaXimus. These designs are divided into three sections: structural features, robot control, and vision. The structural features section describes the unique physical systems of MaXimus. The robot control section describes various feedback controls systems used to maneuver the robot and the relation between the control systems. The vision section describes the method used to parse video to determine the location robots and ball on the soccer field.

2 Structural Features

The BYU ECEn Robot Soccer project requires design of a physical robot. MaXimus was designed to play soccer. For this purpose, many features have been incorporated to improve MaXimus' soccer-playing abilities.

*Undergraduate student, walter@byu.edu.
†Undergraduate student, robbie@byu.edu.

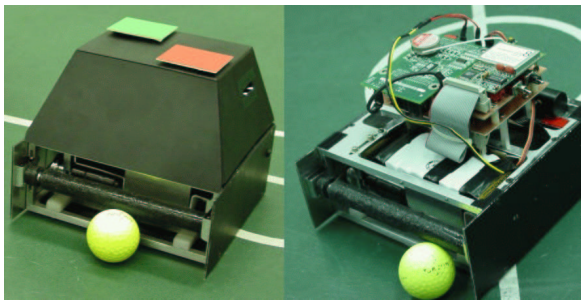

Figure 1: MaXimus, a Robot Soccer senior project.

Figure 2 shows these features implemented in MaXimus. The golf ball, shown in Figure 2, is the ball used in competition and is not part of the robot. The physical design of our robot was done using Pro/E and milled using a CNC. The chassis was milled out of quarter inch aluminum stock. The front and back ends of MaXimus are labeled in Figure 2.

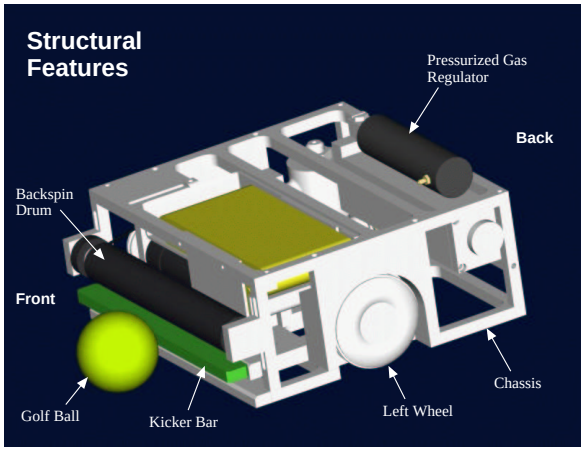

Figure 2: Structural Features of MaXimus.

The kicker bar shown in Figure 2 and Figure 3 gives MaXimus kicking ability. It is made of aluminum and slides in the forward direction to "kick" the ball. A pneumatic piston, shown in Figure 3, actuates the kicker bar. The solenoid gas value, shown in Figure 3, controls the flow of gas into the pneumatic piston. The pressurized gas regulator connects directly to the CO_2 cartridge holder and supplies gas to the solenoid gas valve.

The backspin system helps to retain the ball at MaXimus' front and is shown in Figure 4. When the ball is in contact with the backspin drum, the ball spins and continually rolls toward MaXimus. A motor and pulley supply the backspin drum with rotational velocity. With backspin on the ball, MaXimus is able to make turns or move away from the ball and still retain the ball.

3 Robot Control

This section describes the components of MaXimus' control system. Figure 5 shows the relationship between the low and high level motion controller and the flow of data for the control system. The *overhead camera* faces down onto the field and captures video. The *vision system* parses the captured video and determines global position of the robots and ball. *Artificial intelligence* decides which *play* to run. The running *play* determines the path that the robot should follow. Various *skills*, e.g. feedback linearization, are used to create correctional velocities needed to move the robot along the desired paths. Those desired velocities are sent over

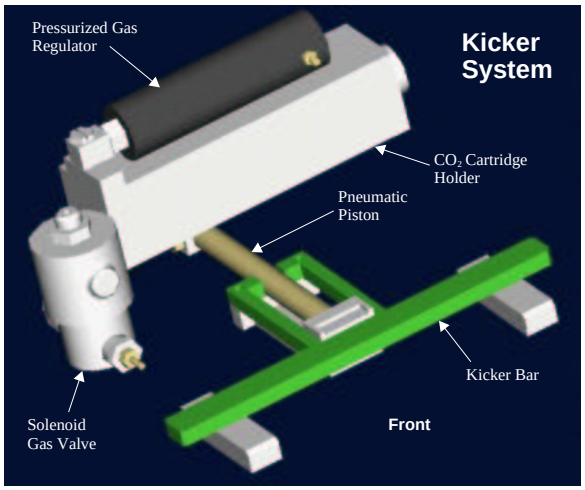

Figure 3: The kicker system of MaXimus.

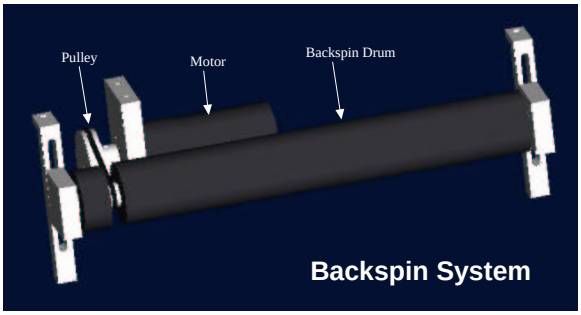

Figure 4: The backspin system of MaXimus.

radio frequency (RF) modems and read by the *microcontroller*. All low-level velocity control is handled by the *microcontroller*, *motor drivers*, *motors*, and *encoders*.

3.1 Low-Level Velocity Controller

The low-level velocity controller is a major contributer to the performance of MaXimus. It resides on the robot and ensures that the angular velocities of the wheels match desired velocities. The velocity controller ran a much tighter feedback control loop than our competitors and with less noise in velocity estimation. This is accomplished by eliminating two problems.

1. Noise introduced by the vision system. Velocity is obtained by approximating the derivative of position. When noise is introduced into the signal, its high-frequency component corrupts the velocity estimation. This noise is introduced into the signals of the vision system.

2. Latency in position update caused by the vision system equipment. Our system is constrained to discrete signals and our velocity estimation is done using differencing methods. That is, velocities are found by dividing the difference in position between samples by the period between samples. As the period between samples increases, the velocity estimation becomes less accurate. Data is available from vision system equipment at a rate of thirty frames a second. We desire a faster rate of sampling.

MaXimus' velocity controller uses a PID feedback loop to control the velocities of the left and right wheels. This feedback loop runs in the micro-controller and custom designed hardware. Figure 6 demonstrates the PID feedback controller used to regulate the angular velocity. The constants K_p, K_i, and K_d are the gain values used to tune the responsiveness of the feedback system.

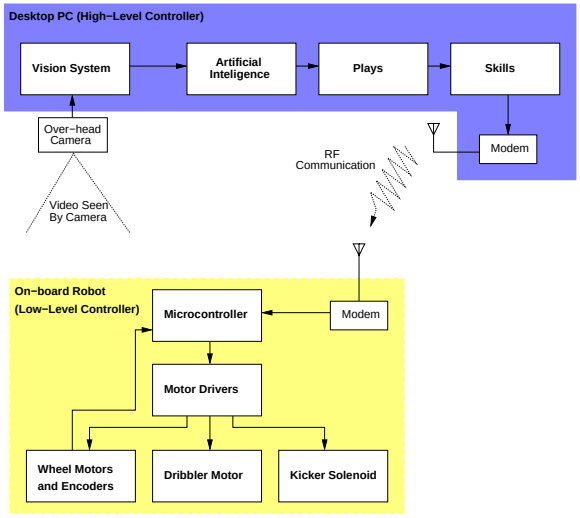

Figure 5: Overview of the signal flow.

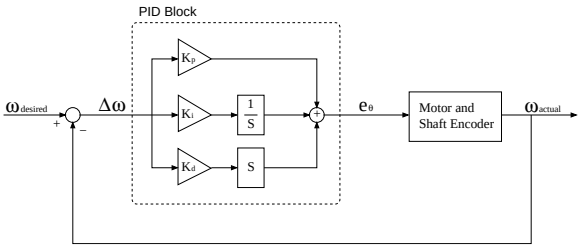

Figure 6: PID feedback controller for the angular velocity.

Error in angular position of the robot, $\Delta\theta$, is interpreted as the desired angular velocity, $\omega_{desired}$. This value, $\omega_{desired}$, is feed through the PID block. The output of the PID block, the control effort, or e_θ, is used to drive the motors. The forward velocity controller uses the same configuration as the angular velocity control with different gain values. Input for the forward velocity controller is $\Delta\psi$ or $v_{desired}$.

Our velocity controller uses DC motors with 512 counts per revolution quadrature encoders. The LS7266R1 encoder to microprocessor interface chip is used to track encoder counts and monitor the motors shaft angles. The micro-controller calculates the motor velocity using the equation

$$\omega_{motor} \approx \frac{\Delta\theta_{motor}}{\Delta t},$$

where ω_{motor} is the motor shaft angular velocity, $\Delta\theta_{motor}$ is the difference in motor shaft angle between the last two samples, and Δt is the sample time period.

The wheel velocity, ω_{wheel}, relates to ω_{motor} by the motor gear head ratio, K_{gear}. The relationship is

$$\omega_{wheel} = \frac{\omega_{motor}}{K_{gear}}.$$

Recognize that there is a motor and wheel on both the left and right right sides. Subsequently, two wheel velocities are calculated. The key words $_{motor}$ and $_{wheel}$ represent either the left or right side.

MaXimus uses a “wheel-chair” design where the wheels are perpendicular to one line that passes near the robots center of gravity (CG). Figure 7 shows a top view of the robots wheel orientation. The wheels are equidistant from the CG. Because of this wheel orientation, it is desirable to place the PID feedback controllers on the angular velocity and forward velocity rather than the individual wheel velocities. Motivation for this

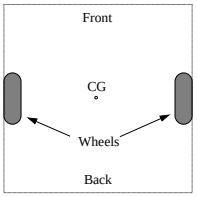

Figure 7: Wheel Orientation.

came by observing the power required for acceleration of the robot in the forward direction as opposed to acceleration in the angular direction.

Figure 8 illustrates the variables that are used in this paper. Note that ω_r and ω_l are the right and left wheels velocities. The variables v and ω represent the robot's forward and angular velocities respectively.

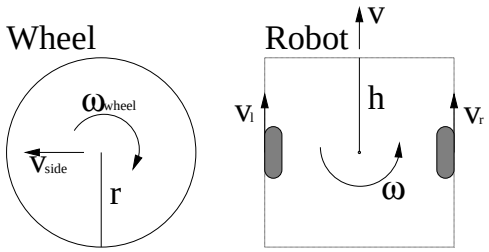

Figure 8: Variable definitions.

To apply the feedback control in this manner, we must translate the actual wheel velocities to forward and angular velocities. This is done using the following matrix transformation:

$$\begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2h} & -\frac{r}{2h} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} = \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{1}$$

where r is the wheel radius and h is the hand length or distance from the center of the robot to the center of the robot's front as shown in Figure 8. The control effort applied to each motor must be translated from forward and angular effort to left and right effort. We first find the inverse of the first matrix in Equation 1:

$$\begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2h} & -\frac{r}{2h} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{r} & \frac{h}{r} \\ \frac{1}{r} & -\frac{h}{r} \end{bmatrix}. \tag{2}$$

Next, find the efforts that are applied to the left and right wheels:

$$\begin{bmatrix} \frac{1}{r} & \frac{h}{r} \\ \frac{1}{r} & -\frac{h}{r} \end{bmatrix} \begin{bmatrix} e_\psi \\ e_\theta \end{bmatrix} = \begin{bmatrix} e_r \\ e_l \end{bmatrix}, \tag{3}$$

where e_r and e_l are the left and right wheel control efforts.

3.2 Position Controller

Feedback linearization is used to translate positional errors in the Cartesian plane, the playing field, to the errors of the forward and angular position of the robot. These errors are used as desired velocities and are fed into the velocity controller. Feedback linearization is ideal for positional control of MaXimus because its location is defined in the Cartesian plane and because of the robot's non-holonomic design. The non-holonomic constraints limit the direction of the robot's velocity vector [7]. MaXimus' non-holonomic properties are due to its "wheel-chair" design.

A hand-point is define at a fixed point relative to the robot. We have defined this point to be at the center of the robot's front side. All reference to position is two dimensional, looking down onto the soccer playing field. Points relative to the playing field are defined in the Cartesian plane. A desired point at which we would like the hand-point is defined at a location on the playing field.

The errors between the hand-point and the desired point, defined by Δx and Δy, are translated to $\Delta\psi$, the error in distance in the robot's forward direction, and $\Delta\theta$, the error in angle of the robot, according to

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{1}{h}\sin\theta & \frac{1}{h}\cos\theta \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \Delta\psi \\ \Delta\theta \end{bmatrix}. \tag{4}$$

The matrix in Equation 4 is the rotation matrix. The variable θ is the robot's angle and h is the length of the hand-point from the center of the robot.

Figure 9 illustrates the feedback linearization controller used in MaXimus. The block labeled "Feedback Linearization" is the transformation matrix of Equation 4. The PID blocks reside in the robot's microcontroller.

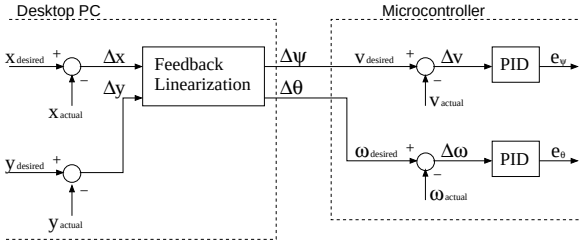

Figure 9: Feedback linearization.

3.3 Path Generation

Path generation allows MaXimus to follow a desired path at a specified velocity and to avoid collision with walls or the opponent. The path generation routine outputs a desired hand-point location which is fed into the position controller. The desired hand-point is always generated at a fixed distance from the robot's actual hand-point. This ensures that the control effort produced by the position controller will be consistent.

In order to understand how the path generation routine works, we will discuss the robot's deviation from a desired *Line* path, where a *Line* path is a line segment that begins and ends at the desired points WP_{start} and WP_{end} (see Figure 10). All points are reference globally. The point R represents the robot's hand-point and is given by the vision system. The point P is the projection of the hand-point onto the path. A gray shaded rectangle identifies common regions in Figures 10 and 11.

We desire a control effort that corrects path deviation proportionally and maintains a constant control effort. To do this, control effort must decrease in the direction of WP_{end} and increase in the direction of P as the robot deviates more from the *Line* path.

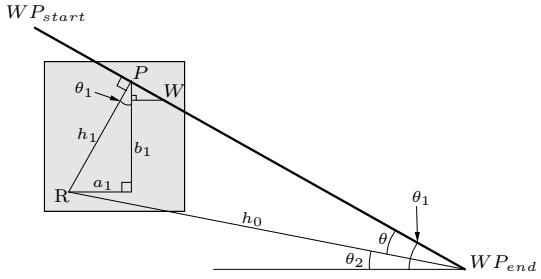

Figure 10: Path generation for a *Line* path.

A directing point, W, is found a distance h_2 from P on the *Line* path and in the direction of WP_{end}. The direction of W is ideal for the robot's hand-point to travel because it directs a deviating robot back onto the *line* path and in the direction of WP_{end}. The point W is not an ideal input for the position controller in that large deviations will result in control saturation. For this reason a second directing point, $W2$, is introduced as shown in Figure 11. The point $W2$ lies a distance h_4 from R in the direction of W. This ensures consistant

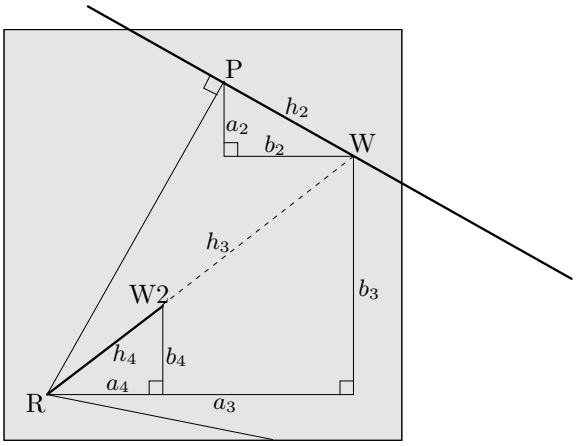

Figure 11: An inlarged view of the path generation method on a *Line* path.

effort regardless of the amount of path deviation. The point $W2$ is the desired hand-point location and is the input of the position controller. Note that h_2 and h_4 are of fixed lengths.

A point's coordinates are denoted by x and y subscripted with the point's variable (i.e. the point R has coordinates x_R and y_R). With the given points R, WP_{end}, and WP_{start} we can find θ_1 and θ_2 as

$$\begin{aligned} \theta_1 &= tan^{-1}(\frac{y_{WP_{start}} - y_{WP_{end}}}{x_{WP_{end}} - x_{WP_{start}}}), \\ \theta_2 &= tan^{-1}(\frac{y_R - y_{WP_{end}}}{y_{WP_{end}} - x_R}). \end{aligned}$$

From Figure 10 we see that $\theta = \theta_1 - \theta_2$ when θ. The sides of the first triangle are calculated as

$$\begin{aligned} h_1 &= h_0 sin\theta, \\ a_1 &= h_1 sin\theta_1, \\ b_1 &= h_1 cos\theta_1. \end{aligned}$$

With a_1 and b_1, we realize the coordinates of the projection point, P, as

$$\begin{aligned} x_P &= x_R + a1, \\ y_P &= y_R + b1. \end{aligned}$$

We find the sides of the second triangle, a_2 and b_2, as

$$\begin{aligned} a_2 &= a_1 \frac{h_2}{h_1}, \\ b_2 &= b_1 \frac{h_2}{h_1}. \end{aligned}$$

Subsequently, the coordinates of W are

$$\begin{aligned} x_W &= x_P - a_2, \\ y_W &= y_P + b_2. \end{aligned}$$

The distance between R and W, represented by h_3 in Figure 11, is found as

$$h_3 = \sqrt{(a_3)^2 + (b_3)^2}. \tag{5}$$

The sides of the forth triangle, a_4 and b_4, are

$$\begin{aligned} a_4 &= a_3 \frac{h_4}{h_3}, \\ b_4 &= b_3 \frac{h_4}{h_3}. \end{aligned}$$

The input of the position controller, $W2$, can be found as

$$x_{W2} = x_R + a_4,$$
$$y_{W2} = y_R + b_4.$$

In addition to *Line* paths, *Arc* paths have been implemented. This architecture can accommodate more complex paths, e.g. *Bezier Curves*. Robot velocity is controlled by scaling gain values. Velocity control of the robot as it follows a desired path is useful for ball handling, collision avoidance, and stability. Note that h_2 and h_4 can be set to different values to change performance so long as $0 < h_4 < h_2$.

After implementation, our team learned that the path generation routine becomes less stable as P approches WP_{end}. It is important to switch input of the position controller from the generated point $W2$ to the actual end of the desired path when the distance between P and the WP_{end} is shorter than h_2.

3.4 Vision

A vision system is used to locate the positions of robots and the ball on the soccer field by parsing video captured by the overhead camera (see Figure 5). A fast, accurate vision system allows for more precise robot control. After researching vision systems of past RoboCup teams, we decided to use a blob detection method implemented by a team from Ngee Ann Polytechnic in Singapore [8]. The idea of this method is to locate the centroid and mass (the total number of pixels contained within the blob) of blobs of color, where a blob is any set of adjacent pixels of the same color. The blobs are categorized and sorted by their relative masses so that blobs of interest may be identified. For example the ball is the only yellow object on the field. Once the yellow blobs have been sorted then the largest yellow blob is assumed to be the blob representing the ball.

The steps of the algorithm are:

1. Locate all runs, where a run is a horizontal line segment composed of pixels of the same color.
2. Combine adjacent runs of the same color into blobs.
3. Calculate the centroid and mass of each blob that exceeds a minimum threshold.

The method for locating runs is similar to that used in run-length-encoding algorithms. Each pixel is determined to be one of five values: red, green, blue, yellow, or black (background). Runs are identified and stored with their starting and ending pixel number, line number, and color.

Blobs are composed of adjacent runs of the same color. A run becomes part of a new blob if the run is on the first line or the run does not match color with any adjacent runs on the previous line. A run is added to an existing blob if the run matches color with an adjacent run on the previous line. As each run is added to a blob, the blobs parameters are updated: its mass and the sum of pixels in the x and y directions. After all blobs are found, the center of each blob of color is determined by dividing the x and y sums by the total mass.

This blob detection method has several advantages over the filter-based methods of our competitors:

1. The algorithm is very tolerant to noise in the data. This allowed us to specify wider color thresholds so that as many pixels as possible could be clamped to a color value. This allowed the locations of dots of color on the field to be more precisely determined. The whole system is also highly tolerant to changes in lighting conditions.
2. The vision system accurately locates individual blobs of color even when blobs of the same color are close together. Each robot has a green dot on top of it. When two robots on the field are close together, separating the two green dots is very difficult unless you can assign each individual pixel to one of the two dots.
3. More than two-thirds of the image data can be thrown away without significantly affecting accuracy of the system. Processing less data allows the system to cycle quicker. This gave us a major advantage as we could process the incoming camera data in less than 15ms per cycle (Fast enough for about 70 cycles per second). While our competitors' vision systems were a major speed bottleneck, our rapid cycling allowed us to devote more processing power to path generation and artificial intelligence routines without compromising reaction time.

4 Summary and Discussion

In this paper we have described fundamental areas of design which were implemented in MaXimus, a Robot Soccer project in the Department of Electrical and Computer Engineering at BYU. A low-level velocity controller regulates the forward and angular velocity of our robot. A position controller generates desired velocities necessary to move the robot to a desired point. Path generation routines allow MaXimus to track tight line and arc paths and ensure controlled effort.

From experience, we learned that our path generation routine is less stable for short paths. To avoiding pathological behavior, input for the position controller must be switched from path generation to the path's endpoint as the robots hand-point nears the end of the path.

Tactical maneuvering should be considered to handle the ball properly and to avoid the opponent. Artificial intelligence is used to control the paths, plays, and unique tricks.

Some of the disadvantages experienced from implementation of designs discussed in this paper are as follows. The on-board velocity controller had PID gain values that required tuning. This was more difficult than tuning values located on the desktop PC. MaXimus' endpoint was at the ball rather than beyond it. This meant that control effort diminished with proximity to the ball. This allowed other robots to push MaXimus around when near the ball.

Gain values in the PID blocks of the velocity controller as well as the distances h_2 and h_4 in the path generation need to be tuned to produce more desirable behavior.

Acknowledgments

In addition to the first author, team MaXimus was composed of two other electrical and computer engineering undergraduate students: Nicolas Bikhazi and Brian Fretz, who all contributed significantly to the project. We gratefully acknowledge the contributions of Matthew Blake and Steve Olson who were TAs for the class.

References

[1] "Robot soccer." http://www.ee.byu.edu/ee/srprojects/robot, May 2002.

[2] M. Asada and H. Kitano, "The robocup challenge," *Robotics and Autonomous Systems*, vol. 29, pp. 3–12, 1999.

[3] "Federation of international robot-soccer association (fira)." http://www.fira.net.

[4] T. Braunl, "Research relevance of mobile robot competitions," *IEEE Robotics and Automation Magazine*, vol. 6, pp. 32–37, December 1999.

[5] R. W. Beard, "Robot soccer: An ideal senior design experience," in *Proceedings of the American Control Conference*, (Chicago, Illinois), pp. 3975–3979, June 2000.

[6] R. W. Beard, J. K. Archibald, and S. A. Olson, "Robot soccer as a culminating design project for undergraduates," in *Proceedings of the American Control Conference*, (Anchorage, Alaska), May 2002.

[7] J.-H. K. Jung-Min Yang, "Sliding mode motion control of nonholonomic mobile robots," *IEEE Control Systems Magazime*, pp. 15–23, April 1999.

[8] "Blob detection." http://www.np.edu.sg/ ˜nbk/, Feb 2002.