A Method Applied for Soccer's Behaviors Using Proper Feedback and Feedforward Control

Hironori Mizuno[1]* and Masakatsu Kourogi[1] and Yukihide Kawamoto[1] and Yoichi Muraoka[1]

School of Science and Engineering,Waseda University, 3-4-1 Okubo Shinjuku Tokyo 169, JAPAN

Abstract. As a practical way of achieving "Athletic Intelligence(AI)", we have tried to build a robot capable of playing soccer. In this paper, we made a mobile robot that can shoot and dribble a ball. In order to shoot the rolling ball, the robot must predict the future position of the ball so that it can move ahead of the ball. That is, the robot should be controlled by feedforward control rather than feedback control because feedback control does not allow enough time to catch the ball. Therefore, we think that it is important that the robot has internal model with which it can predict the target position. As long as the ball is within the field of view, the robot is under feedforward control. At the final stage of shooting, control is switched to feedback to minimum errors. When dribbling the ball through the flags, the robot must move without touching the flags and also keep the ball in front under feedback control. Since the robot has an internal model, it should follow the target using feedback control. We have checked that proper use of both control improves the two athletic movements so the robot must have an internal model that can predict the future state of the target object.

1 Introduction

As a practical way of achieving "Athletic Intelligence(AI)", we have tried to build a robot capable of playing soccer. In this paper, we made a mobile robot, called "Shoobot", that can shoot and dribble a ball. Its skills are measured in terms of the quality of specific tasks.

In this paper, we will discuss the two control problems at the level of the computational model. We especially focus on the issue of which of feedback or feedforward control should be applied to a certain given situation. We define feedforward control as which makes Shoobot move on a trajectory to the position of the target precited by the internal model.

In the shooting problem, there are three steps.

1. The ball is far from Shoobot and thus the distance cannot be measured accurately enough. Since Shoobot cannot predict the trajectory of the ball, it has no choice but to follow the ball; this is FB control.
2. When the ball is within the scope of the visual system of Shoobot, the distance

* http://www.info.waseda.ac.jp/muraoka/members/hiro/

can be obtained accurately enough to allow the robot to predict the future position of the ball. Since the trajectory is obtainable, Shoobot can set one contact point and move ahead of the ball; this is FF control.
3. When the ball is near, Shoobot can be shot at the final stage. Shoobot adjusts its course so that it can position the ball just in front of the shooting device. The second step is crucial for to enable shooting behavior. Without it, Shoobot and the ball will arrive at the contact point at the same time, thus making it difficult to shoot the ball precisely.

In the dribbling problem, Shoobot must satisfy the following two different constraints. 1) Shoobot must keep the ball in front of its body. The local constraint can be satisfied through FB control. 2) Shoobot must move through the flags and finally reach the goal. This global constraint can be satisfied through FF control

It is vital for a soccer-playing robot to have an internal model which enables feedforward control.

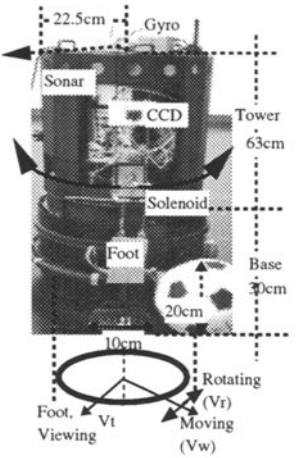

Fig. 1. "Shoobot"

2 Feedback and Feedforward

FB is a method of reducing the error less from the target in the system. FB control is kept under the constraint of the current state and thus does not predict the future state. On the other hand, we define FF control as that which makes Shoobot move on the trajectoy to the target position predicted by the internal model.

Feedforward Control (FF) and Feedback Control (FB)

	Prediction	Following target	Controlling Time	Extra Time
FF	Yes	No	fast	Yes
FB	No	Yes	slow	No

Although, unlike FB control, FF control does not correct error, the predicted state may be updated by new information immediately afterit becomes available. If FF control is performed well, Shoobot's motion is stable. As a result, Shoobot can have extra time to prepare for three next action.

2.1 Mechanism of shooting

In order to shoot a rolling ball, FF control is required so that Shoobot can move ahead of the ball, because FB control does not allow enough time to catch the ball.

Shoobot requires a model that can predict the position of the rolling ball in near future. If Shoobot can always predict the contact point, it can go there smoothly without stopping. That is, the internal model guides Shoobot to where it should go and when it should stop. In this case, Shoobot has the merit that it can move quickly and describe its own trajectory without suffering from the effects of the external state on the way.

Using a CCD camera attached on the body as shown figure 1, we have determined that how each pixel on the captured 2D image corresponds to a position in real space. However, not all pixels correspond to a specific point because the lens is bent at its circumference and a pixel corresponding to a far point becomes relatively longer. (the camera is set to look down), and because the extent of the distance is limited according to the camera's position on the body and its direction. Our single-camera (1.57[rad] wide, with 1/2 CCD, attached at a height of 0.7m) on the tower can distinguish an object at the maximum distance of only about 2[m]. If we can use two cameras, this limit of the distance can be made longer.

If the ball is beyond the limit of the distance, Shoobot cannot move via FF control because the future position of the target cannnot be predicted. We will explain FF and FB control of Shoobot according to the distance.

1. The case that Shoobot cannot determine the distance because the ball is too far (beyond 2[m]): Shoobot moves via FB control by maintaining constant visible angle to the ball. In this case, Shoobot does not use distance information.
2. The case that Shoobot can determine the distance (within 2[m]): Shoobot predicts the future position from past positional data and moves quickly using FF control.
3. The case that the ball is near to Shoobot: After robot arrives at the predicted position, it waits for the rolling ball. In order to shoot the ball with one shot, Shoobot adjusts its position to the minimum distance.

In the second process, Shoobot calculates the speed vector of the ball from past positions and predicts the next position. We assumed that in the internal model, the ball rolls at a constant speed along straight line. If Shoobot's view is separate from its body, that is, captured from a global position [IROS96] such as a camera on the ceiling, Shoobot can use the position information in the entire space. With a global camera, Shoobot does not require separate processes such

as those described above. This means that Shoobot can always predict the target position. In either case, robot must finally catch the ball using FB control.

When robot moves only via FB control, gain in FB control must be adjusted. If this gain is larger, robot can follow the movement of the rolling ball. In reverse, if the gain is small, robot cannot catch up with the rolling ball.

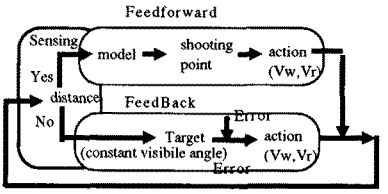

Fig. 2. Flow chart for shooting

2.2 Mechanism of dribbling

We will explain here how the robot dribbles the ball through the flags. In shooting behavior, the most important thing is whether robot can predict the position of the rolling ball and move ahead to the predicted point. In this case, it is not necessary to consider for Shoobot's path.

On the other hand, in dribbling behavior, it is necessary to consider the path, because robot must always keep the ball in front of its body. That is, dribbling behavior must simultaneously satisfy two constraints. One is that robot moves with the ball in front of the body. The other is that robot carries the ball to a target such as a goal.

1. Dynamic change of moving target
   Robot can have only one target to move because robot's path cannot be separated. In the task in which robot must dribble the ball through the flags without touching them. In this case, the next target is decided by the position of the next flag. If next target fixes at one point, Shoobot must change its wheel angle more frequently as Shoobot comes near it, which modification leads to unrequired motion.
2. Dynamic control of the wheel speed to keep the ball in front
   Shoobot must move within practical constraints such as the robot's shape and the ball size. Since both Shoobot and the ball is round, the contact between them may be at only one point. Therefore, Shoobot can push the ball without dragging. If the ball has little friction with the floor, it causes the ball to roll quickly and move around the body unexpectedly. Shoobot must always modify the wheels' direction keeping the ball on the line to the target, which was decided in the first process.

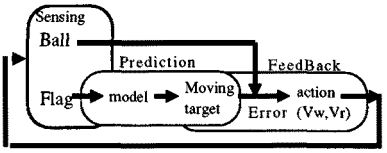

Fig. 3. Flow chart for dribbling

3 Shooting motion

We explain here how to implement the shooting motion on Nomad200.

3.1 Shooting the rolling ball

There is a vital limit in the time to reach the predicted point. In order to shoot the ball, Shoobot should judge whether it can reach the point or not within this time. Forestalling time also depends on the current conditions of the set of the wheel angle and speed. Check whether Shoobot starts moving. When the ball is

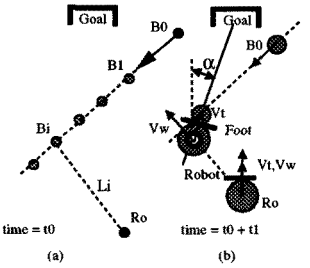

Fig. 4. How to predict the position of the rolling ball

observed moving from B0 to B1, as shown in figure 4, predicted point (Bi) is i times the ball vector in the rolling direction. Each distance (Li) from the present position (Ro) is calculated.

$$\mathbf{B_i} = \mathbf{Forestall}(Internal)(i) + \mathbf{B_0}(i = 1, \ldots, number)$$

For each point, Shoobot compares the time required to move (RT_i) to the rolling time (BT_i). Since Nomad200 has nonholonomic structure as stated in Chapter 3, moving time is calculated as the total time of rotation and moving.

First, the wheels rotate at the maximum speed until the wheel angle becomes angle indicated in the target direction. After that, Shoobot moves at the maximum speed to the goal. If this time is shorter than that required to reach the rolling ball, Shoobot can start to move to the target point. In the reverse case, Shoobot cannot reach the target point before the ball arrives there.

$$RT_i = \frac{L_i}{Max_{RobotSpeed}} + \frac{Rot_i}{Max_{RotSpeed}}$$

$$BT_i = (Interval)(i)$$

$$RT_i > BT_i : Startmoving$$

$$RT_i \leq BT_i : Notmoving$$

3.2 Moving algorithm

Moving time RT_i is defined as the sum of each time at the maximum speed. Since we define rotation time and moving time independently, this sum represents a maximum time as the upper limit. Since Nomad200 can perform rotation and movement of the wheel simultaneously, there is one merit that the total time may be shorter. However, it is difficult to reduce the wheel errors when adjusting both parameters simultaneously. Therefore we have tuned each parameter to attain a balance between the rotation and the moving.

For wheel movement and rotation, 0.83 and 0.36 second respectively, are needed to achieve top speed (Vt: 0.6[m/s],Vr: 1.31[rad/s]). Moving time at the beginning is greater than the rotational one. It is desirable not to alter the moving speed frequently in order for Shoobot to reache the goal faster. We adopted the moving strategy below. If the angle to the next target is larger than a certain value, rotation of the moving direction should proceed rather before forward motion. After the moving direction is set, Shoobot begins to move at the maximum speed. The moving speed is decreased as Shoobot approaches the target. As a result, we found that Shoobot arrives at the goal with an error of less than 10 cm regardless of the wheel angle at the starting point. If the differential angle is less than 0.44 rad, Shoobot winds its way through the global path.

We found that Shoobot can move stable by means of these 3 process: moving to the predicted position of the ball, waiting for the ball using FF control and finally shooting it using FB control.

3.3 Shooting a stationary ball

Shoobot requires an internal model to use FF control. We explain the task to make Shoobot move around the ball. This movement is performed so that wheel direction finally corresponds to the direction of the ball and the goal. Shoobot moves around the ball smoothly by changing the wheel direction, and can finally reach the point behind the ball.

As the internal model [ALIFE96], Shoobot uses the potential field model with repulsive and attractive forces. As explained in 2.1, as long as Shoobot can determine the distance to the ball, Shoobot can move around the ball via FF control.

Imagine a relative coordinate in which the ball is at the origin and the goal is on the x-axis. To generate the path described above, a force should be applied

to Shoobot in such a way that it is pushed toward the -y direction if positioned in Quadrants I and IV and straight to the target if positioned near the ball. Therefore, it is desirable for its path to be heart-shaped when viewed behind the goal.

To produce an path of this shape, we have designed a dynamics model, in which an attractive force and a repulsive force is virtually set on the coordinate defined by the goal and the ball. Figure 5 shows that this model assumes three fixed points: P1 on the positive x-axis which has a repulsive force, P2 on the negative x-axis which has an attractive force, and O on the origin which has a repulsive force less than P2. Combination of these forces generates the moving path of Shoobot. This explains the 3 fixed forces produced at each point (P1,O,P2). **(1)** The constant attractive force +a direction. **(2)** The repulsive force at the origin which is not in inverse proportion to the distance. **(3)** The repulsive force in the -b direction which is in inverse relation with the square of the distance. A point far from the origin is not influenced by force (2) or (3) and

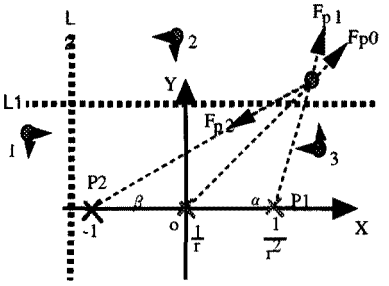

Fig. 5. A field model of repulsive and attractive force

is pulled to point P2 by only force (1). When Shoobot approaches sufficient close to the ball, forces from the origin and P1 begin to strongly push Shoobot away and it is eventually drawn to point P2. The origin represents the position of the ball. The repulsion prevents unwanted collision. In this dynamic model, the functions for the calculation are shown below. Fp1, Fo, Fp2 represent an action force from each point P1,O and P2 respectively. Rp1, Ro and Rp2 represent the distances of P1, O, and P2 respectively to Shoobot. Fx and Fy represent the x- and y-coordinates of the force,respectively.

$$Fx = F_{p1} + \frac{Fo(R_x - a)}{R_o} + \frac{F_{p2}(R_x - b)}{R_{p2}},$$

$$Fy = \quad + \frac{F_o R_y}{R_o} + \frac{F_{p2} R_y}{R_{p2}}$$

$$F_{p1} = c\frac{1}{R_{p1}^2},$$

$$F_o = \frac{1}{R_o},$$
$$F_{p2} = -1$$

After dt time units, Shoobot moves to the new position (Rx,Ry) calculated from vector (Fx,Fy).

$$\frac{dRx}{dt} = Fx,$$
$$\frac{dRy}{dt} = Fy$$

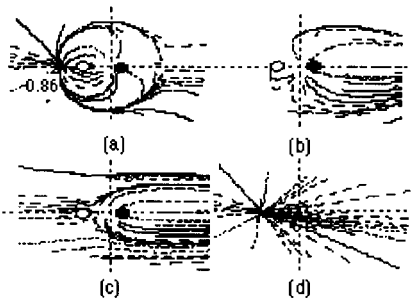

Fig. 6. Comparison of result with 3 different parameters

The dynamics of the path is manipulated via 3 parameters. Black circle marked at P1 and white circle marked at P2, the value (+a) to the length of P1, the value (+b) to the length of P2, the value (+c) represents weight as shown in Figure 5

Figure 6 represents 4 distinctive features of the results of computer simulation. For each case, the parameters are (a) a=0.5, b=0.1 and c=0.05, (b) a=0.5, b=0.1 and c=1.0, (c) a=0.5, b=0.1 and c=3.0, (d) a=10.0, b=1.0 and c=0.2, and illustrate the domain that is limited from -2 to +2. As the value +c increased (a-b-c), the stream of the lines tend to be pulled in the positive x direction. This trend is dependent on the lengths of P1 and P2 and we did not choose the path behind the ball in the case of straight-line motion (d). The shooting algorithm described below uses the same parameters as in case (a). According to the parameters in case (a), Shoobot is pulled down at the point that is 0.86 [m] in the x negative axes.

4 Dribbling the ball

Nomad200[Nomad200] can wind its way using 2DOF (moving forward or backward and turning left and right). The body is divided into the two parts shown

in figure 1. The tower rotates around independently of the basement in which three wheels are equipped. Nomad200's movement ls limited in an action under the nonholonomic structure: it can go forward in only one direction at one time.

While Shoobot is adjusting the wheel angle, the ball is also rolling. It is desirable for appropriate Shoobot movement to shift the target in the internal model to reduce errors. This is achieved through two interactive controls related to the influence of errors of the ball on change of the global target and the effects of the changing target on motion to keep the ball in front.

Shoobot has two standard coordinates on the body. One is virtual in the camera's frame, and the other, called the body coordinate, is such that at the initial time, the wheels' direction is defined as X-axis and the other axes are defined by the left-handed rule. The target can be represented directly as a position on the body coordinate.

The center of gravity of the ball seen in the 2D image corresponds to the position in the 3D body coordinate though this position includes some error. As a result, Shoobot can predict the moving point on the 3D body coordinate on which Shoobot, the ball and the flags exist.

The camera is set to look down and is parallel to the body coordinate so that Shoobot can easily determine the position from visible coordinates. The field of view extends 0.5[m] forward and 0.3[m] to the right and left sides. Unless Shoobot can adequately keep the ball in front, Shoobot will lose sight of the ball (0.22[m] in diameter).

4.1 Dribbling the ball on a line

In this section, we describe how Shoobot can actually dribble the ball on a line. Since Shoobot is bilaterally symmetric, we explain its behavior in the case of rolling, using only one side. We describe how Shoobot moves around the ball on the left side. We have selected the entire path for the case that Shoobot reaches the goal with the minimum distance and time.

In dribbling, we have classified the relationship between the path, Shoobot and the ball into two categories. The 1st category corresponds to the situation where the ball is within the safety zone, and the 2nd category outside the safety zone. Furthermore, the 2nd category is divided into 3 patterns.

We define the safety zone in which Shoobot can push the ball using feedback control. This zone is bounded by two lines drawn away from at a fixed width from the moving direction.

If the ball is in the safety zone, Shoobot has only to push the ball using feedback control. It controls and rotates wheels slightly in order to position the ball on a center line in sight. If the ball is outside the zone, Shoobot can move around the ball with three patterns of movement.

A constant angle from the moving direction is fixed for each side of. The line forming this angle is represented by the dot line in figure 7(a-2). We define the counterclockwise rotation as the positive angle. The following 3 patterns of movement shown in Figure 8, are adopted to bring the ball back within the safety zone.

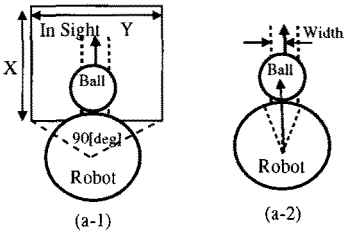

Fig. 7. Controlling the rolling ball

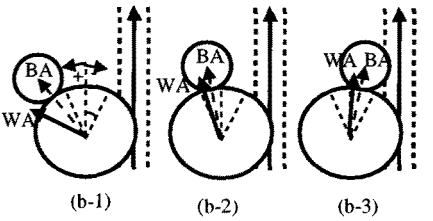

Fig. 8. Three patterns of movemnent

1. The position of the ball is beyond the positive fixed angle. Fig.8(b-1)
   If Shoobot pushes the ball forward, it will immediately lose the ball on the left side. If the speed is slow, wheel angle is dynamically calculated to be larger than the ball angle at which the ball is lost. If Shoobot does not slow down, the rotational speed should be proportionally accelerated. Since the wheels does not rotate at the top speed (0.79 deg/s), it is necessary to reduce the speed.
2. The position of the ball is within the positive fixed angle. Fig.8(b-2)
   When Shoobot pushes the ball forward, it will gradually lose the ball to left side. In this case, the ball must be shifted towards the right. Wheel angle should be set slightly larger than the ball angle.
3. The position of the ball is within the negative fixed angle. Fig.8.(b-3)
   In this case, the ball may gradually roll to the right when feedback control is used. Unless it does not push the ball enough, wheel angle may be set a slightly smaller than the ball angle. With the repetition of these processes, Shoobot gradually nears the global path without confusing the direction of the rolling ball.

4.2 Dribbling the ball through the flags

We explain the task of dribbling the ball through flags and finally shooting the ball into the goal. Shoobot always has only a target on the way because Shoobot can chose the adequate path under multiple targets. It is important not to touch the flags, the next target (T_n) chosen by Shoobot is a point on the line drawn

from the present Shoobot's position to a circle around the next flag. This line to the target (R_nT_n) is defined as the ideal path along which keep the ball. Shoobot sets the new target (T_{n+1},T_{n+2}) using all sensing data and its present position. This corresponds to FF control.

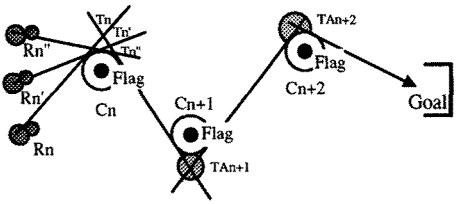

Fig. 9. Plan for dribbling

Merely by pushing the ball, it may deviate from the correct direction. Furthermore, Shoobot must return to a new path using FB control. Where the fixed target is on the path, Shoobot can dribble the ball at the speed of 0.62m per sec. After passing through a flag, Shoobot must decide a new target around the next flag.

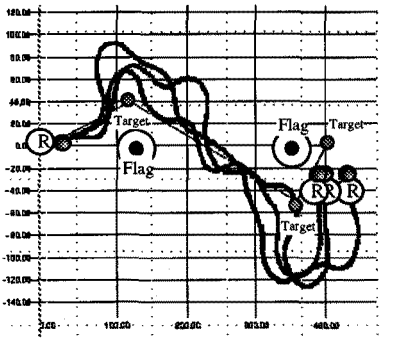

Fig. 10. Paths after dribbling the ball

5 Discussion and Summary

We found that feedforward control is required to determine the position of the space from visible information and to operate the distance and position using an internal model. If no appropreate internal model exists, Shoobot should be controlled to follow the present target by feedback control wherein Shoobot moves while maintaining the ball at a constant angle of view.

An internal model of the environment (used to determine the speed vector of the ball) enables Shoobot to predict the position of contact and to move there before the ball arrives. We have checked that athletic movement is improved througth proper use of FF and FB control.

Bizzi's experimental result [Bizzi84] showed that a monkey's arm moves as if there is an original path to the target in his brain. Kawato suggested that humans control their arms efficiently using FF control with a highly sophisticated internal model [Kawato96]. These results indicate that creatures with brain may make use of internal representation, though the representation may be different from the form of the coordinates we used in our research. [Arbib89]

There is a criticism that the use of representation leads to symbolizing (see [Brooks95]). We support the notion that robots should include internal representation as indicated by the present experiments. With an internal model, the robot's movement becomes more stable and the target can be reached in a shorter time. Furthermore, Shoobot itself can also modify its path by transfering the the force position represented in the internal model such as potential field in section 3.3.

We conclude that the use of an internal model enables Shoobot to set its target and to use feedfoward control. As a result, the robot can move more smoothly.

References

[Brooks95] R. A. Brooks : Intelligence without Reason. The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents, L. Steels, R. Brooks(Ed), 25–81

[ALIFE96] Mizuno H., M. Kourogi, Y. Kuroda, Y. Muraoka : An Architecture for Soccer Shooting Robot. ALIFE V, PP-12, May (1996)

[IROS96] Mizuno H., M. Kourogi, Y. Muraoka : Building "Shoobot" Capable of Dribbling and Shooting. 96IROS, WS4, Pre-RoboCup, (1996)

[Arbib89] Arbib, M. A. : The Metaphorical Brain 2: Neural Networks and Beyond. Wiley-Interscience, (1989)

[Bizzi84] Bizzi. E., Accornero, N., Chapple, W., Hogan, N.: Posture Control and Trajectory Formation during Arm Movement. The Journal of Neuroscience, 4, 2738-2744 (1984)

[Kawato96] Kawato, M. : Bi-Directional Theory Approach to Consciousness. Cognition. Computation and Consciousness. Oxford University Press, Oxford (1996) http://www.erato.atr.co.jp/

[Nomad200] Nomadic Technologies, Nomad200 User's Manual