An Architecture for Action Selection in Robotic Soccer

Peter Stone and David McAllester

AT&T Labs — Research
180 Park Ave.
Florham Park, NJ 07932
{dmac,pstone}@research.att.com
http://www.research.att.com/~{dmac,pstone}

ABSTRACT

CMUnited-99 was the 1999 RoboCup robotic soccer simulator league champion. In the RoboCup-2000 competition, CMUnited-99 was entered again and despite being publicly available for the entire year, it still finished in 4th place. This paper presents some of the key elements behind ATT-CMUnited-2000, one of the three teams that finished ahead of CMUnited-99 in RoboCup-2000 out of thirty four entrants. Playing against CMUnited-99, ATT-CMUnited-2000 scores an average of about 8 goals per opponent goal. This paper describes some of the key innovations that make this improvement possible.

1. INTRODUCTION

RoboCup is a continuing AI research initiative that uses the game of soccer as a unifying and motivating domain [9]. RoboCup as a whole includes technical sessions, robotic demonstrations, competitions with real robots, and a simulator competition. The RoboCup simulator competition, pits teams of 11 independently-controlled autonomous agents against each other in the RoboCup simulator, or Soccer Server, a real-time, dynamic environment [2].

One feature of the RoboCup simulator competition is that many participating teams release their teams, as an executable and/or source code, after each event. Thus, there is typically dramatic progress in the level of play (measured by performance against the previous year's champion) from year to year [4].

CMUnited-99 was the 1999 RoboCup robotic soccer simulator league champion. Unlike the previous champions, it performed quite well in the subsequent competition, RoboCup-2000, finishing in 4th place despite having been publicly available for the entire year. This paper presents some of the key elements behind ATT-CMUnited-2000, one of the three teams that finished ahead of CMUnited-99 in RoboCup-2000. Playing against CMUnited-99, ATT-CMUnited-2000 wins by an average score of 2.5–0.3 in 10-minute games: it scores an average of about 8 goals per opponent goal.

ATT-CMUnited-2000 is based on CMUnited-99 but with a variety of innovations. First, we use an option-evaluation architecture for controlling the player with the ball. In this architecture an option is an object (in the object-oriented programming sense) which can be scored and can be executed. The architecture simply selects the option with the highest score and executes it. The use of objects to represent options allows a fixed architecture to operate over a wide variety of different kinds of options. In ATT-CMUnited-2000 the options include, among others, passing and shooting. The option-evaluation architecture and the general theory of option scoring is described in Section 2.

A second innovation is the introduction of leading passes — the ability to pass the ball at an oblique angle to the intended receiver (as opposed to passing only directly to the receiver). Because we allow leading passes, hundreds of different pass options are considered for the same intended receiver.

A third innovation is the development of an efficient numerical algorithm for computing interception times. The interception time of a given player and a given pass (initial kick) is the time required for that player to reach the ball assuming it continues on its initial path. A pass option is determined by the initial kick — the initial direction and velocity of the passed ball. The intended receiver is simply the teammate with the earliest interception time for that kick. The key opponent is the opponent player with the earliest interception time. The safety margin is the difference between these two interception times. In order to compute safety margins for hundreds of passes it is necessary to do thousands of interception time calculations. While in possession of the ball, e.g., while dribbling, these thousands of interception times are calculated on each cycle of the simulation (100 milliseconds). To make this computation feasible we use a continuous-time modified Newton method which provably converges on the minimum interception time and converges in practice in four or five numerical iterations. This algorithm is described in Section 3.

A fourth innovation is the use of force fields to control the motion of players not with the ball. The force field control is described in Section 4. Related work is discussed in Section 5.

AGENTS'01, May 28-June 1, 2001, Montréal, Quebec, Canada.

2. AN ARCHITECTURE FOR ACTION SELECTION

An option is something that can be scored (evaluated) and executed. In our C++ implementation we have a general class for options with various different subclasses corresponding to different kinds of options for activities such as passing, dribbling, and shooting. When a player has the ball, it considers many options and executes the one with the highest score.

In this architecture it is important that the scores be comparable, e.g., if a certain shooting option has a higher score than a certain passing option then one would hope that the shooting option is really "better". From a theoretical perspective, option scores are best understood as expected rewards in the sense of reinforcement learning [8]. One might say, for example, that a goal achieves a reward of 1000 and an opponent goal gives a reward of -1000. The idealized score of an option can be defined to be the expected reward in, say, the next 30 seconds of play if that option is taken. If one could really compute this idealized score then one would have confidence that higher scoring options are really better. In practice, of course, it is not possible to exactly calculate the expected reward for a given option and in our implementation we use hand-written scoring functions. However, the intended semantics of the score — expected reward — provides conceptual guidance when writing the scoring functions.

We score a pass or shot option by calculating a probability that the option is successful — that the pass is completed or that the shot scores a goal. The score of the option is then of the form $p_s v_s + (1 - p_s) v_f$ where p_s is the probability of success and v_s and v_f are the values of succeeding and failing respectively. Although our value functions are currently hand-written, it would be natural to have agents learn the probabilities p_s and values v_s and v_f as a function of features of the current position. We leave this learning for future work. Automated learning of value functions should make it easier to "calibrate" the values such that, for example, pass values can be meaningfully compared with shot values. This calibration across different kinds of options seems difficult for hand written scoring functions.

2.1 Soft Boolean Expressions

We use hand written functions for scoring options and for computing the force fields described in Section 4. In our experience it seems that continuous functions are preferable to functions with discontinuous steps. We have found that certain primitives are quite useful in writing continuous functions that are easy to understand qualitatively. The basic idea is to replace Boolean expressions with "soft" functions. We do not intend our use of soft or "fuzzy" Boolean expressions to have any deep philosophical significance. We simply want a way of writing easily understood continuous functions.

For example, rather than compute the Boolean value $x < y$ we use $x <^{\delta} y$ which is a real number in the interval $[0,\ 1]$ and is defined as follows.

$$\begin{aligned} x <^{\delta} y &\equiv s((y - x)/\delta) \\ s(x) &\equiv 1/(1 + e^{-x}) \end{aligned}$$

Here $s(x)$ is the sigmoid function — we have $s(x) \in [0,\ 1]$; $s(0) = 1/2$; $s(x) \sim 0$ for $x << 0$; and $s(x) \sim 1$ for $x >> 1$. So if $y - x$ is large compared to δ then $x <^{\delta} y$ is near 1 and if $x - y$ is large compared to δ then $x <^{\delta} y$ is near 0.

We also use a soft or fuzzy version of conditional expressions. In particular, we define the function $\mathtt{if}^*$ as follows.

$$\mathtt{if}^*(p, x, y) \equiv px + (1 - p)y$$

Here we assume $p \in [0,\ 1]$ and for $p \sim 1$ we have $\mathtt{if}^*(p,\ x,\ y) \sim x$ and for $p \sim 0$ we have $\mathtt{if}^*(p,\ x,\ y) \sim y$. We often write expressions of the form $\mathtt{if}^*(x <^{\delta} y,\ z,\ w)$.

2.2 Pass Options

CMUnited-99 players only consider passing the ball directly to teammates at a velocity such that the ball will be moving at a fixed target velocity when it reaches the intended receiver [5]. Thus, there are at most 10 passes to consider at any given time (1 to each teammate). On the other hand, ATT-CMUnited-2000 considers passing the ball in arbitrary directions and at arbitrary velocities. There are potentially infinitely many possible passes.

In practice, ATT-CMUnited-2000 considers passing the ball at discreet angle increments (4^o by default) and speed increments ($0.2 m/sec$ by default). For each angle-speed pair, ATT-CMUnited-2000 begins by determining which teammate and opponent would be quickest to intercept the ball were the ball to be propelled from its current position at the given angle and speed.

Let I_t be the minimum time it would take a teammate to intercept the hypothetical pass, and let I_o be the corresponding time for opponents. Our formula for computing the probability that the pass succeeds is the following.

$$p_s = \mathtt{if}^*(I_t <^5 I_o, .9, 0)$$

That is, the larger the margin between the teammate's and the opponent's interception times, the higher the probability of success for the pass.

The success value v_s is determined by the location at which the teammate would intercept the hypothetical pass. In ATT-CMUnited-2000 v_s is a simple function over ball position that increases as the ball advances towards the opponent's end-line, with strong attraction points at the opponent's goal and the corners adjacent to their goal. Thus, down-field passes are favored, and the team is particularly encouraged to move the ball towards the corners or towards the opponent's goal. For the pass option, v_f is taken to be zero.

The calculations of p_s, v_s and v_f are clearly heuristic. For example, as is described in Section 3, interception time calculations assume that the players move at constant velocity in the ideal direction from the moment the ball is kicked. This assumption can be modified by manipulating the assumed player velocity as well as assuming that players wait a certain amount of time before noticing the ball's trajectory and starting to move towards it. In order to be conservative about passing decisions, ATT-CMUnited-2000 assumes that the opponents can move faster and react more quickly than teammates.

The calculation of v_s is unrealistic in that it considers only the ball's position. In reality, the configuration of players on the field plays a large role in the value of the resulting state after a pass.

Even with these simplifying assumptions, qualitatively ATT-CMUnited-2000 is often able to pass the ball in such a way that teammates can "run on" to it and such that the ball advances towards the opponent's goal when possible, but away from the goal when such a pass has a significantly higher probability of success.

2.3 Shot Options

In addition to considering hundreds of pass options, the player with the ball must also consider the possibility of shooting, particularly when it is near the opponent's goal. A separate option is needed for this action due to the fact that the value of p_s does not depend on teammates being able to intercept the ball. It only relates to whether an opponent can intercept the ball before it reaches the goal.

As such, let I_p be the nearest *point* at which an opponent can intercept the ball. If I_p is off the field (i.e. the ball would have to enter the goal to get to I_p), then let $d = 0$. Otherwise, let d be the distance from I_p to the opponent's goal. Then the success probability p_s is calculated as follows.

$$p_s = 1 - Max(0, Min(1, \frac{d+1}{2}))$$

That is, the closer the opponent's interception point is to the front of the goal, or the further within the goal the interception point, the higher the probability that a goal will be scored. If the opponent could only intercept the ball once it's more than 3 meters within the goal, then we take p_s to be 1. This conservative parameter setting is due to the fact that if the goalie does reach the ball in time, it catches the ball and is freely able to clear it, thus ending the entire goal-scoring opportunity. In general, if there are good passes available, it is best to shoot only when the likelihood of scoring is very high.

The success value v_s for the shot option is a fixed number that is relatively large in the context of the value function used for pass options: a successful goal is the ultimate reward possible in this domain. Again v_f is taken to be zero.

Like in the case for pass options, the player can consider several shot options, each representing a shot to a different part of the goal. The relative merits of the different shot options are easily comparable. However, comparing the relative merits of a shot option and a pass option is not nearly as straightforward. Changing the success value for the shot option, an arbitrary parameter, can drastically affect an agent's selected option.

2.4 Other Options

Due to the difficulty in comparing different types of options, we sought to limit the number of types used, focusing most of the agent's computational resources on evaluating pass (and shot) options. Other option types that we implemented include

Dribble options: A player might consider *dribbling* the ball rather than passing it. Note, however that dribbling can be considered a pass with low velocity such that the kicker is also the receiver.

Hold-ball option: A player can consider standing still with the ball, just keeping it from the opponents and waiting for a passing opportunity to develop.

Clear option: In the defensive zone, it is often useful to simply kick the ball away from the goal such that the opponents can't get it, but without any consideration for whether a teammate can receive the ball.

Send ball option: Similar to the clear option, except that the goal is to get the ball into the opponent's zone (typically towards a corner) rather than away from one's own goal.

Cross option: When a teammate has the ball in the corner, it is often effective to kick the ball hard across the front of the goal (to *cross* the ball). Even though the pass option might evaluate this option unfavorably if several opponents are nearby, the fact that the ball ends up bouncing around in front of the goal can often lead to unpredictable, effective shots.

Although we implemented functions for calculating p_s, v_s and v_f for each of these options, we limited ATT-CMUnited-2000 to considering pass options, shot options, and the clear option. Perhaps more option types could be considered conjunctively provided the values of p_s, v_s and v_f were learned on-line so as to more faithfully represent actual success probabilities and expected rewards.

2.5 Players Without the Ball

A crucial requirement for a soccer team—perhaps more crucial than the decision of what to do with the ball—is an effective mechanism for teammate positioning when not in possession of the ball. Defensive and offensive goals must be balanced, and teammate and opponent actions must be anticipated. ATT-CMUnited-2000's defensive behavior is based mostly on that of CMUnited-99 [5]. However, when on offense, ATT-CMUnited-2000 takes advantage of the keepaway behavior described in Section 4. Under the assumption that the team will be able to generate scoring chances as long as it maintains possession consistently near the opponent goal, at least 5 offensive players, in addition to the one with the ball, move to give the player with ball passing alternatives.

Since we observed that goals are often scored against good teams by crossing the ball across the front of the goal to players in the center, at least two supporting offenders move directly to the front of the goal whenever a teammate gets possession of the ball near an offensive corner of the field. The rest of these offensive players move according to a force-field-based supporting-ball behavior defined in Section 4.

2.6 Results

The resulting team is able to consistently and significantly outperform the RoboCup-99 champion, CMUnited-99. While CMUnited-99 is only able to score about 1 goal every three games against itself (an average final score of roughly 0.3–0.3), ATT-CMUnited-2000 defeated CMUnited-99 by an average score of 2.5–0.3 (roughly an 8–1 margin) over the course of 56 games[1].

Although the competition is not a controlled testing environment, it is also significant to note that ATT-CMUnited-2000 was one of only 3 teams out of 34 to finish ahead of CMUnited-99 at RoboCup-2000, despite the fact that CMUnited-99 was publicly available for almost a full year prior to the event. The fact that the low-level skills used by

[1]The version of ATT-CMUnited-2000 described in this paper is not precisely the one used at RoboCup-2000. The competition version included an additional independent improvement, created by Patrick Riley that improves performance against CMUnited-99 to 2.6–0.2 on average (measured over 33 games). Manuela Veloso was also a co-creator of the overall competition version of ATT-CMUnited-2000.

ATT-CMUnited-2000 are almost identical to those of CMUnited-99 suggests that the action selection architecture described herein is responsible for the improved performance.

3. COMPUTING INTERCEPTION TIMES

In addition to the option-based architecture for action selection, a second innovation embodied in ATT-CMUnited-2000 is an efficient numerical algorithm for computing the time it will take a given player to intercept a moving ball.

As mentioned in the introduction, for a given kick, i.e., initial ball direction and velocity, and a given player (either teammate or opponent), the interception time of that player for that kick is the least time required for that player to reach the ball assuming that the ball is not kicked again in the meantime. In scoring hundreds of pass options the system must compute thousands of interception times. Furthermore, these thousands of interception time calculations are done on each simulation cycle while a teammate is controlling the ball. The implementation can perform tens of thousands of interception time calculations per second.

CMUnited-99 includes a computationally-intensive method of computing interception times based on a discrete time model (one time tick per simulation cycle) and including simulations of the player's discrete turn and run actions. This calculation takes into account the details of the actual soccer server [1, 7]. In CMUnited-99 this interception time calculation is only used for actually going to the ball and is not used for evaluating pass options. For this application, an expensive accurate calculation is appropriate. ATT-CMUnited-2000 also uses the slower, more accurate calculation when actually going to the ball, e.g., when receiving a pass. However, for evaluating pass options a less exact calculation can be used. In evaluating passes ATT-CMUnited-2000 computes interception times assuming that time is continuous and that players can run at a fixed velocity. The details of the numerical algorithm for computing interception times under these assumptions are presented in Appendix A. The algorithm is a modified Newton's method that provably converges to the correct interception time. This convergence is nontrivial since the interception time is the least root of a certain nonlinear function having up to three separate roots. The algorithm for computing interception times and a proof of convergence to the desired root are given in the appendix.

4. FORCE FIELD CONTROL

Another innovation in ATT-CMUnited-2000 is a force-field-based method for controlling players that don't have possession of the ball.

While the team is in control of the ball, the teammates that do not have possession of the ball move in a direction specified by a controlling "force field." The fields used in ATT-CMUnited-2000 were developed for a simplified "keepaway" game. There are two reasons for developing these fields for this simplified game. First, it is much easier to measure performance in a keepaway game than in the full game. Consequently, a hill-climbing approach to program development is more feasible and high performance programs can be developed more rapidly. Second, a team that can hold the ball for extensive periods of time close to the opponent's goal is likely to have more scoring opportunities than the opponents. So good performance in keepaway should translate into good performance on the full game.

Our results on the keepaway task show, not surprisingly, that the motion of players supporting the ball, i.e., offensive teammates other than the one in control of the ball, has an important effect on overall performance. Furthermore, force field control seems to be an effective method of controlling the supporting players.

4.1 The Keepaway Task

For the keepaway experiments, we used a modified version of the RoboCup soccer server [2]. In the keepaway task there is a distinguished offensive team and a distinguished defensive team. The game is played in a series of "trials." At the beginning of a trial, the ball is placed next to the most open offensive player, i.e., the player farthest from the nearest defensive player. The trial lasts until a defensive player gains control of the ball (is within kicking range of the ball for half a second); the ball is passed in a way that violates the offsides rule; or the ball goes out of bounds. When one trial ends a new trial is started by moving the ball to the most open offensive player.

A first objective for the offensive team is to hold the ball for as long as possible, i.e., to make each trial last as long as possible. A second objective is to move the ball as far down-field as possible. In the experiments described here, the players are assigned random positions at the start of the first trial. However, the runs are sufficiently long that performance is dominated by an "equilibrium" player positioning achieved after the first few trials. The keepaway game has no rules other than those ending a trial as described above. When the defensive team (CMUnited-99) gains possession of the ball, it simply holds the ball in order to end the trial, rather than trying to pass and score. Otherwise, the CMUnited-99 team plays as it would in tournament play, which includes trying to take the ball away from the offensive team.

It is possible to use CMUnited-99 as the offensive as well as the defensive team. When playing keepaway against itself, CMUnited-99 has an average possession time (trial length) of about 6 seconds. Many variants of our new program for playing keepaway achieve an average possession time against the CMUnited team of about 25 seconds. One of these variants achieves an average possession time of about 25 seconds and an average ball position about 25 meters from the opponent's end of the field.

4.2 The Basic Keepaway Program

The players in the keepaway experiments are built using CMUnited-99 agent skills [5] as a basis. In particular, their skills include the following:

HoldBall(): Remain stationary while keeping possession of the ball in a position that is as far away from the opponents as possible.

PassBall(t): Kick the ball directly towards teammate t.

GoToBall(): Intercept a moving ball or move directly towards a stationary ball.

In each of the keepaway programs described here, each offensive player is always in one of three modes: "with-ball", "going-to-ball", or "supporting-ball". The player is in with-ball mode if it is within kicking distance of the ball. If no offensive player is within kicking distance then the offensive player that can reach the ball the soonest (as determined by a CMUnited-99 primitive) is in going-to-ball mode. Since each player is actually run by a separate process, each player must decide separately what mode it is in. Because of sensing errors, occasionally two players will both think they can each reach the ball soonest and both go into going-to-ball mode. But this is rare and one can generally think of mode assignment as being centrally determined.

In all of the keepaway teams described here, the with-ball player either executes HoldBall() or PassBall(). When a pass is kicked, the receiver generally becomes the player which can reach the ball the soonest and automatically goes into going-to-ball mode. The player in going-to-ball mode executes GoToBall(): its behavior is identical to that of the CMUnited-99 players in this mode.

In the experiments presented in Section 4.3 the with-ball player is controlled with a somewhat elaborate heuristic. However, based on our experience with controlling the with-ball player, we believe that this elaborate heuristic achieves roughly the same performance as always passing the ball immediately and selecting the receiver that maximizes the minimum angle between the pass and a defensive player no further from the ball than the intended receiver. In the experiments described here we hold the with-ball behavior fixed so that all of the performance differences we observe are a result of differing behaviors of the players in supporting-ball mode.

In all versions of the program described here, the movements of the supporting-ball players are controlled by force fields — each supporting-ball player moves in the direction of a sum of vector fields. Players are kept in bounds with a field that repels the players from the out of bounds lines. This bounds-repellent fields becomes infinitely strong as a player approaches an out-of-bounds line. More specifically, the bounds-repellent field is defined as follows where B_x and B_y are the x and y coordinates of the field, x and y are the player's current x and y coordinates, and $x_{\min}$, $x_{\max}$, $y_{\min}$ and $y_{\max}$ define the in-bounds region.

$$\begin{aligned} B_x &= 5/(x - x_{\min}) - 5/(x_{\max} - x) \\ B_y &= 5/(y - y_{\min}) - 5/(y_{\max} - y) \end{aligned}$$

In general we arrange that a given field will tend to dominate other fields if it has a magnitude large compared to 1. The constant 5 in the above equation causes the out-of-bounds field to become strong if a player is within five meters of the edge of the playing field. At ten meters or further from any edge the bounds-repellent field is weak.

There is also an offsides-repellent field that operates much like the bounds-repellent field to keep players onsides. This offsides-repellent field acts only on the x coordinate of the player and is defined as follows where O_x is the x coordinate of the force field and x_{off} is the x coordinate of the offsides line.

$$O_x \equiv \texttt{if}^*((x_{\text{off}} - x) <^1 5,\ -5,\ 0)$$

This fairly complex formula expresses a rather simple idea. If the player is significantly less than five meters from the offsides line then the force field pushes the player away with a force of five. If the player is significantly more than five meters from the offsides line then the force field is negligible. The field varies continuously from a negligible value to a value near 5 as the player crosses a line five meters from the offsides line.

In general, we decided to try to make all fields continuous functions of the game situation and many of our fields are written in terms of soft Boolean expressions such as $x <^\delta y$ and $\texttt{if}^*(p,\ x,\ y)$ as described in Section 2.1.

In addition to the bounds-repellent and offsides-repellent force fields, there are force fields between players. For a given offensive player, there is a strategic inter-player force due to teammate i, denoted S_i, and defined as follows where d_i is the distance (in meters) to teammate i and U_i is the unit vector pointing in the direction to teammate i (all from the perspective of a player calculating forces on itself due to its teammates).

$$\begin{aligned} S_i &\equiv [(d_i =^{10} 20) - 2(d_i <^{10} 20)]U_i \\ (x =^\delta y) &\equiv e^{-(x-y)^2/\delta^2} \end{aligned}$$

The above is a limited range force field — the strategic force is negligible when significantly further away than 20 meters. The strategic force is attractive between players slightly more than 20 meters apart and repulsive for players closer than 20 meters apart. The basic idea is that players should be within passing distance of each other but far enough apart so that a pass between them would move the ball a significant distance. Note that S_i is a continuous function of d_i and U_i.

Players near the ball are influenced by two tactical inter-player force fields. The first, T_i, is a purely repulsive force between the offensive players. The second tactical force field, the get-clear force, denoted C, pushes a potential receiver away from defenders. The force T_i is defined as follows where again d_i is the distance to teammate i and U_i is the unit vector in the direction of teammate i.

$$T_i \equiv \texttt{if}^*(d_i <^3 8,\ -5,\ 0)U_i$$

Note that T_i is again a continuous function of d_i and U_i. The get-clear force is calculated by first selecting a "key defender" — the defensive player most likely to intercept a pass from the current pall position. The get-clear force is in the direction orthogonal to the line from the ball to the key defender. The strength of the force is governed by an estimate of the probability that the key defender would actually intercept the pass. The precise magnitude and direction for the get-clear force is somewhat complex and could probably be simplified without influencing the performance of the program. We do not present it here. It is worth noting, however, that since the choice of the key defender can suddenly switch as the game configuration changes, the get-clear force can suddenly switch directions. Attempts to make the get-clear force continuous appeared to degrade performance.

Intuitively, the strategic forces apply to players far from the ball and the tactical forces apply to players near the ball. The shift from "near" to "far" is done smoothly. The overall force on a supporting-ball player, denoted F is defined as follows where S is the sum over teammates i of S_i, T is the sum over teammates i of T_i, and d_b is the distance of the player from the ball.

$$F \equiv B + O + \texttt{if}^*(d_b <^{10} 20,\ T + C,\ S)$$

The components of F are illustrated in Figure 1.

A supporting-ball player always tries to run, as fast as possible, in the direction of the combined force F. If it is not currently facing in a direction sufficiently near the direction of F then the player turns toward the direction of

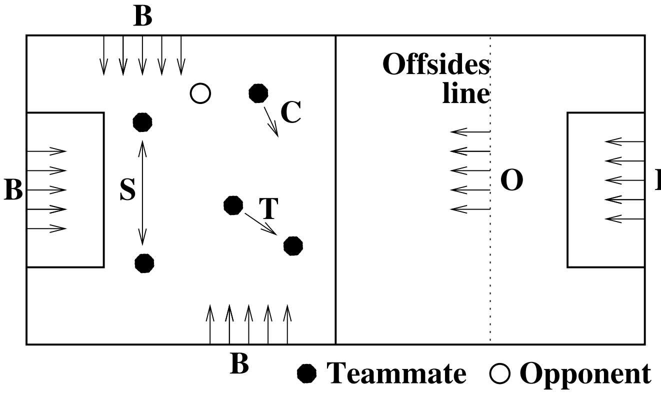

Figure 1: The component forces of F.

Program	Possession Time	Mean Ball x Position
CMUnited	5.7-6.6	-19.5
S	16.9-18.7	-33.6
$S + S^b$	24.8-27.9	-35.9
$S + S^d$	22.2-25.2	25.7
$S + S^b + S^d$	23.7-26.8	26.6

Table 1: Offensive possession time and average x position of the ball when the offensive team is CMUnited and four variations of the basic program. The defensive team is CMUnited in all cases.

F. It turns out that when F is smaller, its direction changes more rapidly and the player spends more time turning and less time running.

4.3 Variations on the Basic Program

Here we consider two additional strategic force fields for controlling the supporting-ball players. The toward-ball strategic force S^b is a force of unit magnitude directly toward the ball. This force pushes supporting-ball players that are far from the ball toward the ball. The forces repelling players from each other, S and T, keep them from bunching up around the ball. The down-field strategic force, S^d, is a force of unit magnitude directly toward the opponents end of the field. In all of the variations of the program considered here, the total field controlling a supporting-ball player has the following form where the strategic field S^* is one of the fields, S, $S + S^b$, $S + S^d$ or $S + S^b + S^d$.

$$F \equiv B + O + \texttt{if}^*(d_b <^{10} 20,\ T + C,\ S^*)$$

The possession time and average x position of the ball for CMUnited and the four variations of the basic program are shown in table 1. The possession time is given as a "95% confidence interval" defined by the mean possession time over a sequence of trials plus or minus $2\sigma/\sqrt{n}$ where σ is the observed standard deviation of the possession time of a trial and n is the number of trials in the run. Since the trials are not really independent, these intervals should perhaps be somewhat wider.

The results show that the choice of fields controlling the supporting players can have a dramatic effect on performance. Furthermore, force field control seems effective on the keepaway task. Our full game implementation divides supporting players into those moving directly to the front of the goal in anticipation of receiving a cross, and those providing "generic support". The generic support players are controlled by the field $S + S^b + S^d$.

5. RELATED WORK

Previous research has explored action generation via vector sums. For example, the Samba control architecture [3] uses two behavior layers: the reactive layer which defines action maps from sensory input to actuator output; and the task layer which selects from among the action maps. In the robotic soccer application, a vector sum of action maps is used to determine the player's actual motion. In this case, the vector sum is not of forces, but of low-level actions.

A previous force-field approach considering sums of attractive and repulsive forces among players and the ball is called strategic positioning using attraction and repulsion, or SPAR [10]. In contrast to our work reported here, these forces were only active over limited regions of the field, and boundaries, such as out-of-bounds and offsides, were treated as hard constraints. SPAR was implemented both in simulation and on real robots.

A different variant of the keepaway task, involving just 3 offenders and 2 defenders, was reported in [6]. In that case, the actions of the offenders were successfully learned via reinforcement learning.

6. CONCLUSION

Innovations in ATT-CMUnited-2000 include

- a general method for considering and selecting from among options when in possession of the ball;
- the ability to pass the ball at an oblique angle to the intended receiver (as opposed to passing only directly to the receiver);
- an efficient numerical algorithm for computing the time required for a given player to intercept a ball at a given position and velocity; and
- force-fields for governing player motion when they are not in possession of the ball.

These innovations enable ATT-CMUnited-2000 to improve significantly over the performance of the team on which it is based, CMUnited-99. While the action selection architecture played a big role in this improvement, it required a tedious manual parameter-tuning phase.

The architecture was motivated by a desire to facilitate reinforcement learning over a larger, more flexible action space than has previously been considered in the simulated robotic soccer task [4]. Our future research agenda includes the exploration of on-line reinforcement learning techniques for combining several different types of action options in a consistent manner using this novel action-selection architecture.

7. REFERENCES

[1] H.-D. Burkhard, M. Hannebauer, and J. Wendler. AT Humboldt — development, practice and theory. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 357–372. Springer Verlag, Berlin, 1998.

[2] I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.

[3] J. Riekki and J. Roening. Playing soccer by modifying and combining primitive reactions. In H. Kitano, editor, *RoboCup-97: Robot Soccer World Cup I*, pages 74–87. Springer Verlag, Berlin, 1998.

[4] P. Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer.* MIT Press, 2000.

[5] P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 35–48. Springer Verlag, Berlin, 2000.

[6] P. Stone, R. S. Sutton, and S. Singh. Reinforcement learning for 3 vs. 2 keepaway. In P. Stone, T. Balch, and G. Kraetszchmar, editors, *RoboCup-2000: Robot Soccer World Cup IV*, Berlin, 2001. Springer Verlag. To appear.

[7] P. Stone, M. Veloso, and P. Riley. The CMUnited-98 champion simulator team. In M. Asada and H. Kitano, editors, *RoboCup-98: Robot Soccer World Cup II.* Springer Verlag, Berlin, 1999.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge,Massachusetts, 1998.

[9] M. Veloso, E. Pagello, and H. Kitano, editors. *RoboCup-99: Robot Soccer World Cup III.* Springer Verlag, Berlin, 2000.

[10] M. Veloso, P. Stone, and M. Bowling. Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer. In *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, Boston, September 1999.

APPENDIX

A. INTERCEPTION TIME CALCULATIONS

In this section we detail an efficient numerical algorithm for computing the time it will take an agent to intercept a moving target assuming that time is continuous and that the agent moves at a fixed velocity. We prove that the algorithm converges to the correct interception time.

For an arbitrary ball position B_0, initial ball velocity V_0 and receiver position R_0, we define the interception time $I(B_0, V_0, R_0)$ to be the least time required for the receiver to reach the ball assuming the receiver can run at a fixed velocity of V_R starting immediately and that the ball continues in its current direction with instantaneous velocity after time t from the initial position given by $V = V_0 e^{-t/\tau}$. The receiver velocity V_r is taken to be the maximum player velocity in the simulator — $1m/s$. The velocity decay parameter τ is adjusted to match the soccer server at the discrete simulation times — $\tau \approx 2s$. This definition of $I(B_0, V_0, R_0)$ abstracts away from the discrete time nature of the soccer server and also abstracts away from the turning and acceleration time of the receiver. However, this abstraction allows $I(B_0, V_0, R_0)$ to be calculated efficiently using relatively simple floating point numerical methods. Furthermore, it is possible to compensate in various heuristic ways for the turning and acceleration time required by the receiver.

To calculate $I(B_0, V_0, R_0)$ we first note that position of the ball after time t, which we will denote as $P(t)$, can be written as follows.

$$P(t) = B_0 + V_0 \tau (1 - e^{-t/\tau}) \tag{1}$$

The interception time satisfies the condition that the distance the receiver can travel during the interception time equals the distance between the initial receiver position and the ball position at the interception time. This condition can be written as follows where $||X||$ denotes the length of vector X.

$$V_r t = ||R_0 - P(t)|| \tag{2}$$

We wish to calculate the least value of t satisfying condition (2). The basic idea is to use Newton's method to find a root of (2). The problem is that condition (2) often has three different roots. More specifically, for a fast moving ball passing nearby there is an earliest and a latest time the ball can be reached as it passes. There is also the earliest time the ball can be reached if we let it pass and then catch up with it as it slows down. All three of these times satisfying condition (2) and we want to find the earliest time satisfying the condition.

Our numerical algorithm is a Modification of Newton's method in which we start with a value of t^0 and iteratively compute t^{i+1} from t^i. Our rule for computing t^{i+1} from t^i ensures that if t_0 is no larger than the interception time then the infinite sequence t^0, t^1, t^2, ... is monotonically increasing and converges to the interception time, i.e., the least solution to (2). In practice, however, we can stop the iteration as soon as the two distances in equation (2) agree to within, say, a tenth of a meter. In general, this termination condition will be reached within three or four iterations — the calculation is quite fast and can therefore be used many times per simulation cycle.

We set t^0 to be the d/V_r where d is the distance from R_0 to the line defined by the point B_0 and the vector V_0. Clearly the receiver cannot reach the ball any sooner than the time required for the receiver to reach the line on which the ball travels.

To formulate the problem in a way appropriate for Newton's method we first define $g(t)$ as follows.

$$g(t) = ||P(t) - R_0|| - V_r t \tag{3}$$

Now condition (2) can be written as $g(t) = 0$. Note that $g(t)$ is the distance between the receiver and the ball at time t if the receiver were to run directly toward the point $P(t)$. The standard Newton's method update sets t^{i+1} to be s^{i+1} as defined by the following equation.

$$s^{i+1} = t^i - g(t^i)/g'(t^i) \tag{4}$$

Note that if $g(t^i)$ is positive and $g(t)$ is decreasing as t increases then the derivative $g'(t^i)$ is negative and this update yields $s^{i+1} \geq t^i$. Unfortunately, for the particular function g under consideration, it is possible that there is a local minimum of $g(t)$ prior to the desired interception time. Past this local minimum it is possible that both $g(t^i)$ and $g'(t^i)$ are positive in which case the update (4) yields $s^{i+1} < t^i$. If we always set t^{i+1} equal to s^{i+1} we might oscillate about this non-root local minimum. If the interception occurs as the receiver is overtaking a ball that is slowing down, it is also possible for the standard Newton iteration defined by (4) to overshoot, i.e., we get an s^{i+1} larger than the desired interception time.

Before giving a modified update with the desired convergence properties we first give an expression for the derivative $g'(t)$. Differentiating (3) yields the following where $U(t)$ is the unit vector in the direction from R_0 to $P(t)$.

$$\begin{aligned} g'(t) &= \frac{P'(t) \cdot (P(t) - R_0)}{||P(t) - R_0||} - V_r \\ &= (e^{-t/\tau} V_0 \cdot U(t) - V_r) \end{aligned} \tag{5}$$

If $||P(t) - R_0||$ is zero then the ball has reached the initial position of the receiver. This cannot happen before the interception time. Hence, before interception we can assume that $||P(t) - R_0||$ is nonzero and hence $U(t)$ is a well defined unit vector. We can now rewrite equation (4) as follows.

$$s^{i+1} = t^i + g(t^i)/(V_r - e^{-t^i/\tau} V_0 \cdot U(t^i)) \tag{6}$$

Our update rule is the following where s^{i+1} is defined by equation (6).

$$t^{i+1} = \begin{cases} t^i + g(t^i)/(V_r - e^{-t^i/\tau} V_0 \cdot U(t^i)) & \text{if } V_0 \cdot U(t) < 0 \\ t^i + g(t^i)/(V_r - e^{-t^i/\tau} V_0 \cdot U(s^{i+1})) & \text{if } V_0 \cdot U(t) > 0 \text{ and } g'(t^i) < 0 \\ t^i + g(t^i)/V_r & \text{otherwise} \end{cases}$$

We now prove that $t^{i+1} \geq t^i$, $t^i \leq I(B_0,\ V_0,\ R_0)$, and t^i converges to $I(B_0,\ V_0,\ R_0)$. If V_0 equals zero then we take the last condition in the update equation which computes the interception time in a single step (all further iterations do nothing). So we can now assume without loss of generality that V_0 is nonzero. Let $R_\perp$ be the point on the line on which the ball is traveling that is nearest to the initial receiver position R_0. Recall that the time t^0 is the time required for the receiver to move from R_0 to $R_\perp$. First we consider the case where the ball has not yet reached $R_\perp$ at the time t^0. This occurs if and only if $V_0 \cdot U(t_0) < 0$. In this case the receiver can reach the ball before it reaches $R_\perp$. As long as the ball has not passed the $R_\perp$ the quantity $V_0 \cdot U(t)$ is negative and adds to magnitude (absolute value) of $g'(t)$. Furthermore, in this case the magnitude of the term $V_0 \cdot U(t)$ is getting smaller with time. This implies that, when the interception point is prior to $R_\perp$ the magnitude of $g'(t)$ is decreasing with time over the region preceding the root but is never smaller than V_r. When the magnitude of the derivative is decreasing as we move toward the root, but is still bounded away from zero, Newton's method has all the desired properties mentioned above. When the interception point is before $R_\perp$ the update rule always uses the first clause and corresponds exactly to the classical Newton's method and hence all the desired properties hold.

If $V_0 \cdot U(t^0) = 0$ then the receiver and the ball reach $R_\perp$ at the same time. In this case t^0 is the interception time and we are done. Now suppose that $V_0 \cdot U(t^0) > 0$. In this case the ball is past the point $R_\perp$ by the time it takes the receiver to reach $R_\perp$. In this case the interception point is beyond $R_\perp$ and, intuitively, the receiver must catch up with the ball as it slows down. In this case $V_0 \cdot U(t) > 0$ is an invariant of the updates and the derivative $g'(t)$ is always greater than $-V_r$. So in the case where the ball is past $R_\perp$ at t^0 the third clause of the update rule can never overshoot the interception time. Furthermore, if we always used the third update rule we would converge on the interception time. This is because the steps are always positive and, since we never passed the interception time, the step size must approach zero. But the step size can only approach zero if $g(t)$ approaches zero so we must approach the interception time. The second clause always takes a step as least as large as the last clause. So as long as the second clause never overshoots the interception time, the sequence must converge on the interception time. Note that for $V_0 \cdot U(t) > 0$, this quantity monotonically increases with time. The fact that $V_0 \cdot U(t)$ is increasing implies that in the case where the second clause is selected we have $t^{i+1} \leq s^{i+1}$. Furthermore, for $t \in [t^i, s^{i+1}]$ we have the following.

$$-g'(t) \geq V_r - e^{-t^i/\tau} V_0 \cdot U(s^{i+1}) \tag{7}$$

Since equation (7) bounds $g'(t)$ over the interval of the update, we have that the update of the second clause cannot overshoot the root.