systems[2].

Prior to conducting the research reported in this chapter, we used neural networks to teach an agent to shoot a moving ball into a goal in a simulator based closely upon the Dynasim system [Sahota 96]. The simulator models the non-holonomic wheeled Dynamite robots [Sahota *et al.* 95]. In this scenario, we were able to use one neural network to time the agent's approach towards the ball, and another to aim the ball. The learned behavior successfully enabled the agent to redirect a moving ball with varying speeds and trajectories into specific parts of the goal. By carefully choosing the input representation to the neural networks so that they would generalize as much as possible, the agent was able to use the learned behavior in all quadrants of the field even though it was trained in a single quadrant [Stone and Veloso 98d].

Learning was applied in both simulated and real robotic, to hit a stationary ball into the goal using the concept of learning from easy missions [Asada *et al.* 94a]. By first placing the ball directly in front of the goal and then gradually moving it farther away, thus making the task harder, the learning time was reduced from exponential to linear in the size of the state space within an RL framework. The robots achieved 70% success in simulation and 40% success in the real world.

Another early example of learning an individual skill in the RoboCup soccer server involved using a neural network to enable an agent to learn when to shoot and when to pass [Matsubara *et al.* 96]. An agent was given the ball near the opponent's goal with a goaltender blocking the way and a teammate nearby. Based on the positions of the ball, the goaltender, and the teammate, the agent with the ball learned when it was best to shoot directly at the goal and when it was best to pass.

In a similar setup, using the same Dynasim-based simulator mentioned above, we used memory-based learning to allow an agent to learn when to shoot and when to pass the ball [Stone and Veloso 96a]. This learning was of an individual skill since the goaltender and the teammate used deterministic, fixed behaviors.

[2] A full discussion of related robotic soccer systems, including more recent learning approaches, appears in Chapter 9 (Section 9.2).

Appendix A

List of Acronyms

Following is the list of acronyms used in this thesis.

AAAI: American Association of Artificial Intelligence

AI: Artificial Intelligence

BDI: Belief/Desires/Intentions

DAG: Directed Acyclic Graph

DAI: Distributed Artificial Intelligence

DT: Decision Tree

GA: Genetic Algorithm

GPGP: Generalized Partial Global Planning

ICMAS: International Conference on Multi-Agent Systems

IJCAI: International Joint Conference on Artificial Intelligence

IROS: International Conference on Intelligent Robotic Systems

KIF: Knowledge Interchange Format

KQML: Knowledge Query and Manipulation Language

MAS: Multi-Agent Systems

MiroSot: Micro-Robot Soccer Tournament

ML: Machine Learning

NN: Neural Network

OPL: Only Play Left

OPR: Only Play Right

PLOS: Predictive, Locally Optimal Skills

PRW: Prefer Right Wing

PTS: Periodic Team Synchronization

RCF: Receiver Choice Function

RL: Reinforcement Learning

RMM: Recursive Modeling Method

SPAR: Strategic Positioning using Attraction and Repulsion

TPOT-RL: Team-Partitioned, Opaque-Transition Reinforcement Learning

Appendix B

Robotic Soccer Agent Skills

This appendix complements the description of the CMUnited implementations described in Chapter 3. The implementation details of the simulated and robotic agent skills are presented. In the context of the team member agent architecture, the skills are low-level external behaviors culminating in action primitives.

B.1 CMUnited-98 Simulator Agent Skills

The skills available to CMUnited-98 players include kicking, dribbling, ball interception, goaltending, defending, and clearing. The common thread among these skills is that they are all *predictive, locally optimal skills* (PLOS). They take into account predicted world states as well as predicted effects of future actions in order to determine the optimal primitive action from a local perspective, both in time and in space.

Even though the skills are predictive, the agent *commits* to only one action during each cycle. When the time comes to act again, the situation is completely reevaluated. If the world is close to the anticipated configuration, then the agent will stay in the same behavior mode and carry on with the same skill, acting similarly to the way it predicted on previous cycles. However, if the world is significantly different, the agent will arrive at a new sequence of actions rather than being committed to a previous plan. Again, it will only execute the first step in the new sequence.

Many of the skills presented here were developed in collaboration with Riley [Stone *et al.* 99].

B.1.1 Kicking

Recall from Section 2.2 that an agent can only kick the ball when it is within the `kickable_area` = `player_size` + `ball_size` + `kickable_margin` = 1.885 with the current server parameters.

As a first level of abstraction when dealing with the ball, all reasoning is done as a desired trajectory for the ball for the next cycle. Before a kick is actually sent to the server, the difference between the ball's current velocity and the ball's desired velocity is used to

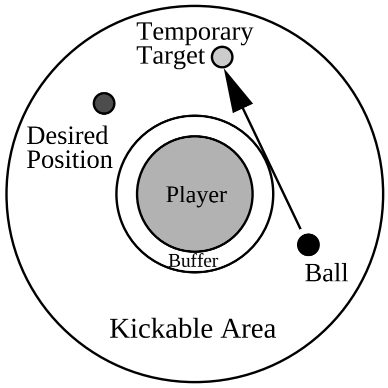


Figure B.1: The turnball skill.

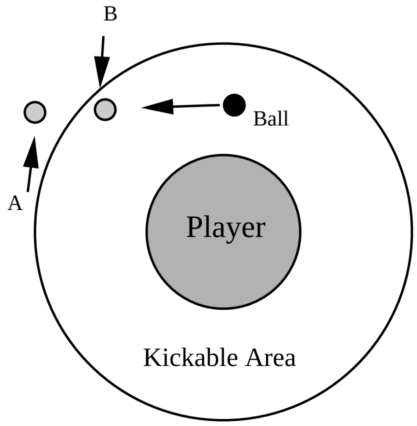


Figure B.2: Basic kicking with velocity prediction.

determine the kick to actually perform. If the exact trajectory can not be obtained, the ball is kicked such that the direction is correct, even if the speed is not.

In order to effectively control the ball, a player must be able to kick the ball in any direction. In order to do so, the player must be able to move the ball from one side of its body to the other without the ball colliding with the player. This behavior is called the *turnball* behavior. It was developed based on code released by the PaSo'97 team [Pagello *et al.* 98]. The desired trajectory of a turnball kick is calculated by getting the ray from the ball's current position that is tangent to a circle around the player (see Figure B.1). Note that there are two possible such rays which correspond to the two directions that the ball can be turned around the player. Care is taken to ensure that the ball stays well within the kickable area from kick to kick so that the player keeps control of the ball.

The next important skill is the ability to kick the ball in a given direction, either for passing or shooting. The first step is to figure out the target speed of the ball. If the agent is shooting, the target speed is the maximum ball speed, but for a pass, it might be better to kick the ball slower so that the receiving agent can intercept the ball more easily. In this case, the agent must take into account the ball's deceleration over time when determining how hard to kick the ball.

In order to get the ball to the desired speed, several kicks in succession are usually required. By putting the ball to the side of the player (relative to the desired direction of the kick) the agent can kick the ball several times in succession. If a higher ball speed is desired, the agent can use the turnball kicks to back the ball up so that enough kicks can be performed to accelerate the ball.

This skill is predictive in that it looks at future velocities of the ball given different possible kicks. In some cases, doing a weaker kick one cycle may keep the ball in the kickable area so that another kick can be executed the following cycle. In Figure B.2, the agent must choose between two possible kicks. Kicking the ball to position A will result in the ball not being kickable next cycle; if the ball is already moving quickly enough, this action may be correct. However, a kick to position B followed by a kick during the next cycle may result in a higher overall speed. Short term velocity prediction is the key to these decisions.

B.1.2 Dribbling

Dribbling is the skill which allows the player to move down the field while keeping the ball close to the player the entire time. The basic idea is fairly simple: alternate kicks and dashes so that after one of each, the ball is still close to the player.

Every cycle, the agent looks to see that if it dashes this cycle, the ball will be in its kickable area (and not be a collision) at the next cycle. If so, then the agent dashes, otherwise it kicks. A kick is always performed assuming that on the next cycle, the agent will dash. As an argument, the low-level dribbling code takes the angle relative to the direction of travel at which the player should aim the ball (see Figure B.3). This is called the "dribble angle" and its valid values are $[-90, 90]$. Deciding what the dribble angle should be is discussed in Section B.1.3.

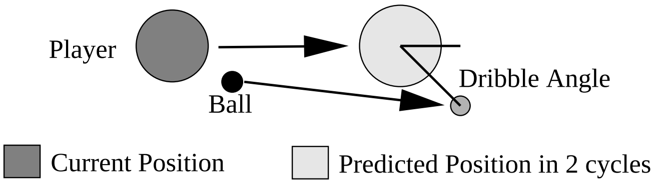


Figure B.3: The basic dribbling skill.

First the predicted position of the agent (in 2 cycles) is calculated:

$$p_{new} = p_{current} + v + (v * pdecay + a)$$

where p_{new} is the predicted player position, $p_{current}$ is the current position of the player, v is the current velocity of the player, *pdecay* is the server parameter `player_decay`, and a is the acceleration that a dash gives. As presented in Section 2.2, the a value is usually just the dash power times the `dash_power_rate` in the direction the player is facing, but stamina may need to be taken into account.

Added to p_{new} is a vector in the direction of the dribble angle and length such that the ball is in the kickable area. This is the target position p_{target} of the ball. Then the agent gets the desired ball trajectory by the following formula:

$$traj = \frac{p_{target} - p_{ball}}{1 + bdecay}$$

where $traj$ is the target trajectory of the ball, p_{ball} is the current ball position, and *bdecay* is the server parameter `ball_decay`. This process is illustrated in Figure B.3.

If for some reason this kick can not be done (it would be a collision for example), then a turnball kick is done to get the ball in the right position. Then the next cycle, a normal dribble kick should work.

As can be seen from these calculations, the basic dribbling is highly predictive of the positions and velocities of the ball and player. It is also quite local in that it only looks 2 cycles ahead and recomputes the best action every cycle.

B.1.3 Smart Dribbling

The basic dribbling takes one parameter that was mentioned above: the dribble angle. Smart dribbling is a skill layered on the basic dribbling skill that decides the best dribble angle based on opponent positions. Intuitively, the agent should keep the ball away from the opponents, so that if an opponent is on the left, the ball is kept on the right, and vice versa.

The agent considers all nearby opponents that it knows about. Each opponent is given a "vote" about what the dribble angle should be; each opponent votes for the valid angle $[-90, 90]$ that is farthest from itself. For example, an opponent at 45 degrees, would vote for -90, while an opponent at -120 degrees would vote for 60. Each opponent's vote is weighted by the distance and angle relative to the direction of motion. Closer opponents and opponents more in front of the agent are given more weight (see Figure B.4).

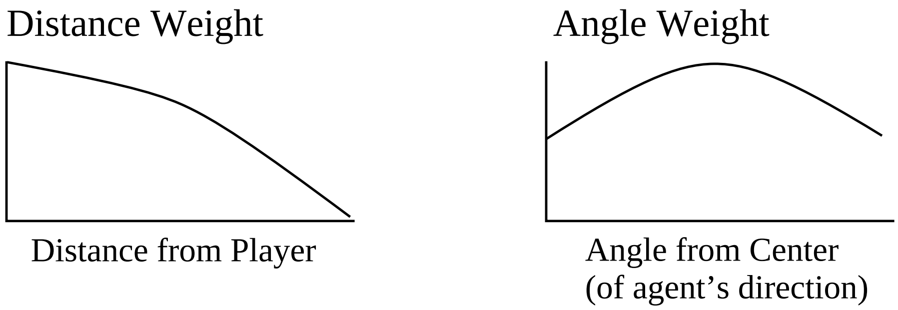


Figure B.4: The weights for smart dribbling.

B.1.4 Ball Interception

There are two types of ball interception, referred to as active and passive interception. The passive interception is used only by the goaltender in some particular cases, while the rest of the team uses only the active interception. Each cycle, the interception target is recomputed so that the most up to date information about the world is used.

The active interception is similar to the one used by the Humboldt '97 team [Burkhard *et al.* 98]. The active interception predicts the ball's position on successive cycles, and then tries to predict whether the player will be able to make it to that spot before the ball does, taking into account stamina and the direction that the player is facing. The agent aims for the earliest such spot.

This process can be used for teammates as well as for the agent itself. Thus, the agent can determine which player should go for the ball, and whether it can get there before the opponents do.

The passive interception is much more geometric. The agent determines the closest point along the ball's current trajectory that is within the field. By prediction based on the ball's velocity, the agent decides whether it can make it to that point before the ball. If so, then the agent runs towards that point.

These analytic ball-interception skills were developed in a later version of the simulator (version 4) than the learned ball-interception skill described in Chapter 5 (version 2). These routines were not appropriate in version 2 of the simulator because players were not given information about the ball's instantaneous velocity in their visual information. In addition, sensory information arrived much less frequently (every 500 msec as opposed to the current 150 msec). Therefore, in version 2 of the soccer server, it was much harder to accurately predict the ball's future path.

B.1.5 Goaltending

The assumption behind the movement of the goaltender is that the worst thing that could happen to the goaltender is to lose sight of the ball. The sooner the goaltender sees a shot coming, the greater chance it has of preventing a goal. Therefore, the goaltender generally uses the widest view mode (`view_width` = *wide*) and uses backwards dashing when appropriate to keep the ball in view while positioning itself in situations that are not time-critical.

Every cycle that the ball is in the defensive zone, the goaltender looks to see if the ball is in the midst of a shot. It does this by extending the ray of the ball's position and velocity and intersecting that with the baseline of the field. If the intersection point is in the goaltender box and the ball has sufficient velocity to get there, the ball is considered to be a shot (though special care is used if an opponent can kick the ball this cycle). Using the passive interception if possible (see Section B.1.4), the goaltender tries to get in the path of the ball and then run at the ball to grab it. In this way, if the goaltender misses a catch or kick, the ball may still collide with the goaltender and thus be stopped.

When there is no shot coming, the goaltender positions itself in anticipation of a future shot. Based on the angle of the ball relative to the goal, the goaltender picks a spot in the goal to guard; call this the "guard point." The further the ball is to the side of the field, the further the goaltender guards to that side. Then, a rectangle is computed that shrinks as the ball gets closer (though it never shrinks smaller than the goaltender box). The line from the guard point to the ball's current position is intersected with the rectangle, and that is the desired position of the goaltender.

B.1.6 Defending

CMUnited-98 agents are equipped with two different defending modes: opponent tracking and opponent marking. In both cases, a particular opponent player is selected as the target against which to defend. This opponent can either be selected individually or as a defensive unit via communication (the latter is the case in CMUnited-98).

In either case, the agent defends against this player by observing its position over time and position itself strategically so as to minimize its usefulness to the other team. When *tracking*, the agent stays between the opponent and the goal at a generous distance, thus blocking potential shots. When *marking*, the agent stays close to the opponent on the ball-opponent-goal angle bisector, making it difficult for the opponent to receive passes and shoot towards the goal. Defensive marking and tracking positions are illustrated in Figure B.5.

When marking and tracking, it is important for the agent to have accurate knowledge about the positions of both the ball and the opponent (although the ball position is not strictly relevant for tracking, it is used for the decision of whether or not to be tracking).

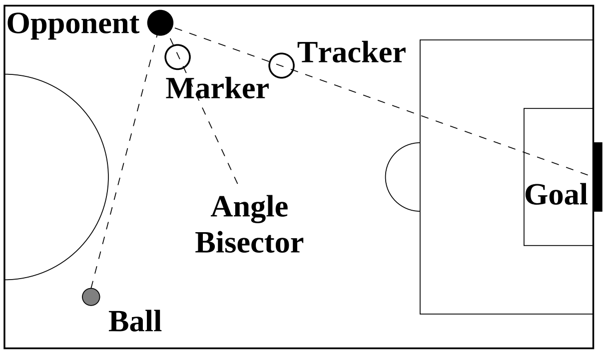


Figure B.5: Positioning for defensive tracking and marking.

Thus, when in the correct defensive position, the agent always turns to look at the object (opponent or ball) in which it is least confident of the correct position. The complete algorithm, which results in the behavior of doggedly following a particular opponent and glancing back and forth between the opponent and ball, is as follows:

- If the ball position is unknown, look for the ball.
- Else, if the opponent position is unknown, look for the opponent.
- Else, if not in the correct defensive position, move to that position.
- else, look towards the object, ball or opponent, which has been seen less recently (lower confidence value).

This defensive behavior is locally optimal in that it defends according to the opponent's current position, following it around rather than predicting its future location. However, in both cases, the defensive positioning is chosen in anticipation of the opponent's future possible actions, i.e. receiving a pass or shooting.

B.1.7 Clearing

When in a defensive position, it is often advantageous to just send the ball upfield, clearing it from the defensive zone. If the agent decides that it cannot pass or dribble while in a defensive situation, it clears the ball. The important decision in clearing the ball is where to clear it to. The best clears are upfield, but not to the middle of the field (you don't want to center the ball for the opponents), and also away from the opponents.

The actual calculation is as follows. Every angle is evaluated with respect to its usefulness, and the expected degree of success. The usefulness is a sine curve with a maximum of 1 at 30 degrees, .5 at 90 degrees, and 0 at -90, where a negative angle is towards the middle of the field. The actual equation is (Θ is in degrees):

$$\text{usefulness}(\Theta) = \frac{sin(\frac{3}{2}\Theta + 45) + 1}{2} \tag{B.1}$$

The expected degree of success is evaluated by looking at an isosceles triangle with one vertex where the ball is, and congruent sides extending in the direction of the target being evaluated. For each opponent in the triangle, its distance from the center line of the triangle is divided by the distance from the player on that line. For opponent C in Figure B.6, these values are w and d respectively. The expected success is the product of all these quotients. In Figure B.6, opponent A would not affect the calculation, being outside the triangle, while opponent B would lower the expected success to 0, since it is on the potential clear line ($w = 0$).

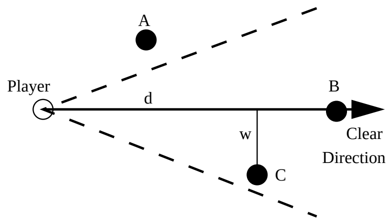


Figure B.6: Measuring the expected success of a clear.

By multiplying the usefulness and expected success together for each possible clear angle, and taking the maximum, the agent gets a crude approximation to maximizing the expected utility of a clear.

Closely related to clearing, there is another behavior called offensive "sending." Rather than trying to clear the ball to the sides, the agent sends the ball to the middle of the offensive zone in the hope that a teammate will catch up to the ball before the defenders. This is useful if the agent is too tired or unable to dribble for some reason. It is especially useful to beat an offsides trap because it generally requires the defenders to run back to get the ball.

The only difference between offensive sending and defensive clearing is the usefulness function. For sending, the usefulness function is linear, with slope determined by the agent's y position on the field. The closer the agent is to the sideline, the steeper the slope, and the more that it favors sending to the middle of the field.

B.2 CMUnited-97 Small-Robot Skills

The skills available to the CMUnited-97 robots include path-planning, two forms of ball handling, obstacle avoidance, goaltending, and pass evaluation [Veloso and Stone 98].

B.2.1 Non-holonomic Path Generation

The non-holonomic path planning problem has been addressed by many researchers, e.g., [Latombe 91, Fujimura 91]. However, most of the algorithms deal with static worlds and generate pre-planned global paths. In the robot soccer domain, this pre-planning is not possible as the domain is inherently dynamic and response times need to be very fast. Furthermore, the world dynamics include possible interference from other robots (e.g., pushing), making precisely mapped out paths ineffective and unnecessary.

Implemented by Han and described in detail in [Han and Veloso 98], the navigational movement for the CMUnited-97 robots is done via reactive control. The control rules were inspired by the Braitenburg vehicle [Braitenburg 84].

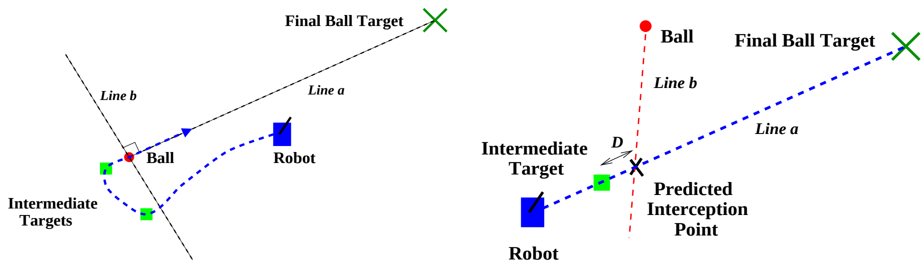
B.2.2 Ball Handling

If a robot is to accurately direct the ball towards a target position, it must be able to approach the ball from a specified direction. Using the ball prediction from the vision system, the robot aims at a point on the far side of the target position. The robots are equipped with two methods of doing so:

- **Ball collection:** Moving behind a ball and knocking it towards the target.

- **Ball interception:** Waiting for the ball to cross its path and then intercepting the moving ball towards the target.

When using the ball collection behavior, the robot considers a line (line a in Figure B.7(a)) from the target position to the ball's current or predicted position, depending on whether or not the ball is moving. The robot then plans a path to a point on line a and behind the ball such that it does not hit the ball on the way and such that it ends up facing the target position. Finally, the robot accelerates to the target. Figure B.7(a) illustrates this behavior.



(a) (b)

Figure B.7: Single-agent behaviors to enable team collaboration (a) Ball collection (aiming for a pass or to the goal); (b) Ball interception (receiving a pass).

When using the ball interception behavior (Figure B.7(b)), on the other hand, the robot considers a line from *itself* to the target position (line a) and determines where the ball's path (line b) will intersect this line. The robot then positions itself along this line so that it will be able to accelerate to the point of intersection at the same time that the ball arrives.

In practice, the robot chooses from between its two ball handling routines based on whether the ball will eventually cross its path at a point such that the robot could intercept it towards the goal. Thus, the robot gives precedence to the ball interception routine, only using ball collection when necessary. When using ball collection, it actually aims at the ball's predicted location a fixed time in the future so as to eventually position itself in a place from which it can intercept the ball towards the goal.

B.2.3 Obstacle Avoidance

In the robotic soccer field, there are often obstacles between the robot and its goal location. Our robots try to avoid collisions by planning a path around the obstacles. Due to the highly dynamic nature of this domain, our obstacle avoidance algorithm uses closed-loop control by which the robots continually replan their goal positions around obstacles. In the event that an obstacle blocks the direct path to the goal location, the robot aims to one side of the obstacle until it is in a position such that it can move directly to its original goal. Rather than planning the entire path to the goal location at once, the robot just looks ahead to the first obstacle in its way under the assumption that other robots are continually moving around. Using the reactive control described above, the robot continually reevaluates its target position. For an illustration, see Figure B.8.

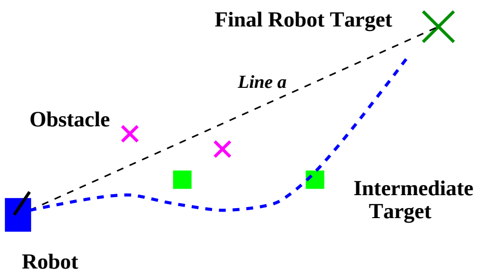


Figure B.8: Obstacle avoidance through dynamic generation of intermediate targets.

The robot starts by trying to go straight towards its final target along line a. When it comes across an obstacle within a certain distance of itself and of line a, it aims at an intermediate target to the side, and slightly beyond the obstacle. The robot goes around the obstacle the short way, unless it is at the edge of the field. Using reactive control, the robot continually recomputes line a until the obstacle is no longer in its path. As it comes across further obstacles, it aims at additional intermediate targets until it obtains an unobstructed path to the final target.

Even with obstacle avoidance in place, the robots can occasionally get stuck against other robots or against the wall. Particularly if opponent robots do not use obstacle avoidance, collisions are inevitable. When unable to move, our robots identify the source of the problem as the closest obstacle and "unstick" themselves by moving away. In order to prevent looping, they move a considerable distance from the closest obstacle before considering themselves unstuck. Once free, normal control resumes.

B.2.4 Goaltending

The goaltender robot has both special hardware and special software. Thus, it does not switch positions or active modes like the others. The goaltender's physical frame is distinct from that of the other robots in that it is as long as allowed under the RoboCup-97 rules (18cm) so as to block as much of the goal as possible. The goaltender's role is to prevent the ball from entering the goal. It stays parallel to and close to the goal, aiming always to be directly even with the ball's lateral coordinate on the field.

Ideally, simply staying even with the ball would guarantee that the ball would never get past the goaltender. However, since the robots cannot accelerate as fast as the ball can, it would be possible to defeat such a behavior. Therefore, the goaltender continually monitors the ball's trajectory. In some cases it moves to the ball's predicted destination point ahead of time. The decision of when to move to the predicted ball position is both crucial and difficult, as illustrated in Figure B.9. Our goaltender robot currently takes into account the predicted velocity and direction of the ball to select its moves. If the ball is moving slowly, it tries to stay even with the ball as shown by the solid goaltender robot; if the ball is moving quickly, it goes directly to its predicted position as shown by the dashed goaltender robot.

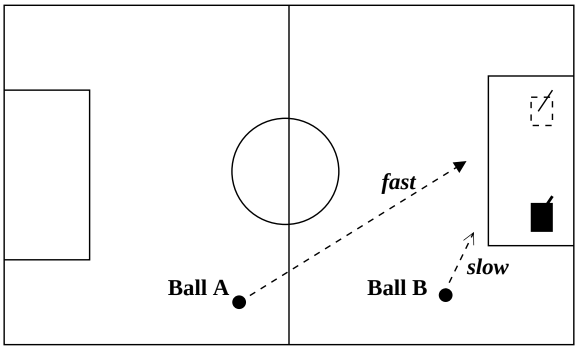


Figure B.9: Goaltending.

B.2.5 Run-time Evaluation of Collaborative Opportunities

Unlike the simulated agents which use a learned pass evaluation function (see Chapter 6), the robots do not have any learned modules. Instead, they use a special-purpose pass-evaluation procedure.

When in active mode, the robots use an evaluation function that takes into account teammate and opponent positions to determine whether to pass the ball or whether to shoot. In particular, as part of the formation definition, each position has a set of positions to which it considers passing (see Chapter 3). For example, a defender might consider passing to any forward or midfielder, while a forward would consider passing to other forwards, but not backwards to a midfielder or defender.

For each such position that is occupied by a teammate, the robot evaluates the pass to that position as well as evaluating its own shot. To evaluate each possible pass, the robot computes the *obstruction-free-index* of the two line segments that the ball must traverse if the receiver is to shoot the ball (lines b and c in Figure B.10). In the case of a shot, only one line segment must be considered (line a). The *value* of each possible pass or shot is the product of the relevant obstruction-free-indices. Robots can be biased towards passing or shooting by further multiplying the values by a factor determined by the relative proximities of the active robot and the potential receivers to the goal. The robot chooses the pass or shot with the maximum value. The obstruction-free-index of line segment l is computed by the following algorithm (variable names correspond to those in Figure B.10):

1. *obstruction-free-index* $= 1$.

2. For each opponent O:

- Compute the distance x from O to l and the distance y along l to l's origin, i.e. the end at which the ball will be kicked by the robot (See Figure B.10).
- Define constants *min-dist* and *max-denominator*. Opponents farther than *min-dist* from l are not considered. When discounting *obstruction-free-index* in the next step, the y distance is never considered to be larger than *max-denominator*. For example, in Figure B.10, the opponent near the goal would be evaluated with $y =$ *max-denominator*, rather than its actual distance from the ball. The reasoning is that beyond distance *max-denominator*, the opponent has enough time to block the ball: the extra distance is no longer useful.
- if $x <$ *min-dist* and $x < y$,
  obstruction-free-index $=$ *obstruction-free-index* $\times$ $x/MIN($*max-demoninator*,$y)$.

3. return *obstruction-free-index*.

Thus the obstruction-free-index reflects how easily an opponent could intercept the pass or the subsequent shot. The closer the opponent is to the line and the farther it is from the ball, the better chance it has of intercepting the ball.

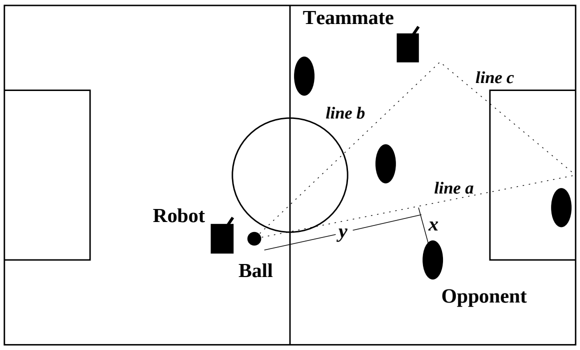


Figure B.10: Run-time pass evaluation based on the positions of opponents.

Appendix C

CMUnited-98 Simulator Team Behavior Modes

This appendix describes the detailed conditions and effects of the behavior modes used by the CMUnited-98 simulator agents. As laid out in Chapter 3, specifically in Section 3.5.1, the agents have several behavior modes. In the context of the team member agent architecture, the choice of behavior mode is the top-level external behavior. The full set of modes is:

- Goaltend
- Localize
- Before kick off
- Face ball
- Set-play
- Recover
- Get onside
- Handle ball
- Active offense
- Auxiliary offense
- Passive offense
- Active defense
- Auxiliary defense
- Passive defense

Here I present the detailed conditions and effects of these behavior modes.

C.1 Conditions

During each simulator cycle, an agent's first step towards choosing an action is to reevaluate its behavior mode. They use the following rule-based system. Some of the conditions depend on an evaluation of which team currently has *possession* of the ball. The agent evaluates which of all the players on the field could intercept the ball's path most quickly. If it is an opponent, then the opponent team has possession. Otherwise, the agent's own team has possession.

The agent uses the first of the following rules that applies.

- If playing the goaltender position: use goaltend mode.
- If own location is unknown: use localize mode.
- If the server's play mode is "before kickoff:" use before kickoff mode.
- If the ball's location is unknown: use face ball mode.
- If there is a set-play defined for the situation and playing a position involved in the play: use set-play mode.
- If there is a set-play defined for the situation and *not* playing a position involved in the play: use passive offense mode.
- If the ball is close enough to kick (within `kickable_area`): use handle ball mode.
- If stamina is lower than the server's threshold for decrementing the `recovery` parameter (`recover_dec_thr` × `stamina_max`) : use recover mode.
- If the ball is not moving and no teammate is closer to the ball: use active offense mode.
- If no teammate could intercept the ball's current trajectory more quickly (taking into account what is known about own and teammates' current staminas): use active offense mode.
- If able to intercept the ball's current trajectory, the opponents have possession of the ball, and no more than one teammate is closer to the ball: use active defense mode.
- If expecting to receive a pass (a teammate has recently announced an intention to execute such a pass): use face ball mode.
- If in an offside position: use get onside mode.
- If playing a defender position:
  - If own team is in possession of the ball and the ball is further back on the field than own position: use auxiliary offense mode.
  - If recently assigned to mark an opponent: use auxiliary defense mode.
  - Otherwise: use passive defense mode.

- If playing a midfielder position:
  - If the opponents have possession: use passive offense mode.
  - Otherwise: use auxiliary offense mode.
- Otherwise (if playing a forward position):
  - If the ball's location is close to the home location of the position being played (within 30 meters on the field's lengthwise axis): use auxiliary offense mode.
  - Otherwise: use passive offense mode

C.2 Effects

The effects of being in the different behavior modes are as follows:

Goaltend: Execute the goaltender skill detailed in Appendix B.

Localize: Turn until visual information is sufficient to determine own location on the field.

Before kick off: Move to a location appropriate for kickoffs (on own side of the field and close to own position's home location). Then face the ball.

Face ball: If the ball's location is unknown, turn until it is visible. Otherwise, turn directly towards the ball.

Set-play: Fill the appropriate role in the appropriate set-play for the current situation. There are defensive and offensive set-plays defined for all dead-ball situations (kickoffs, corner kicks, free kicks, etc.). The number of players involved in each set play varies from 1 (1 player gets in front of the ball when it's the opponent's kick in) to 5 (see the corner kick example in Section 3.5.2). Agents use the current team formation to map formation positions to set-play positions based on geographical proximity. As set-play positions have behaviors associated with them, the agent involved in a set-play executes its assigned behavior either until the play is over or until a time-limit expires. Then it returns to choosing from among the other behavior modes as usual.

Recover: Stay still until stamina recovers above the server's threshold for decrementing the `recovery` parameter: (`recover_dec_thr` $\times$ `stamina_max`).

Get onside: Move to a position that is not offside, i.e. in front of the opponent's last defender.

Handle ball: One of the most important decisions in the robotic soccer domain arises when the agent has control of the ball. In this state, it has the options of dribbling the ball in any direction, passing to any teammate, shooting the ball, clearing the ball, or simply controlling the ball.

In CMUnited-98, the agent uses a complex heuristic decision mechanism, incorporating a machine learning module, to choose its action. The best teammate to receive a potential pass (called *potential receiver* below) is determined by a decision tree trained off-line (see Chapter 6). Following is a rough sketch of the decision-making process without all of the parametric details. The referenced low-level skills are described in Section B.1.

To begin with, since kicks (i.e. shots, passes, and clears) can take several cycles to complete (Section B.1.1), the agent remembers the goal of a previously started kick and continues executing it. When no kick is in progress (do the first that applies):

- If close to the opponent's goal and no defenders are blocking the path to the goal (defined as a cone with vertex at the ball): shoot or dribble based on the goaltender's position, the position of the closest opponent, and the distance to the goal.
- At the other extreme, if close to the agent's own goal and there is an opponent nearby: clear the ball (see Section B.1.7).
- If approaching the line of the last opponent defender: dribble the ball forward if possible; otherwise send the ball past the defender (see Section B.1.7).
- If the potential receiver is closer to the goal and has a clear shot: pass to the potential receiver.
- If no opponents are in the direct path to the goal: dribble to the goal (see Section B.1.2).
- If fairly close to the opponent's goal and there is at most one opponent in front of the goal: shoot.
- If no opponents are in the way of one of the corner flags: dribble towards the corner flag.
- If there is a potential receiver: pass.
- If it's possible to hold onto the ball without moving (at most one opponent is nearby): hold the ball.
- Otherwise: Kick the ball away (clear).

Active offense: Intercept the ball as described in Section B.1.4.

Auxiliary offense: Use the SPAR variant of flexible positioning to try to get open to receive a pass (see Section 3.5.2).

Passive offense: Use the ball-dependent variant of flexible positioning (see Section 3.5.2).

Active defense: Go towards the ball even though another teammate might also be going towards it. As the second player to the ball, this player tries to block the opponent from dribbling away from the first teammate to the ball.

Auxiliary defense: Mark or track an opponent as described in Section B.1.6. The defensive unit captain assigns defenders their opponents to mark so that each opponent is covered.

Passive defense: Use the ball-dependent variant of flexible positioning (see Section 3.5.2).

Appendix D

CMUnited Simulator Team Source Code

The source code of the CMUnited simulator teams is an on-line appendix. It is located at the following URL:

http://www.cs.cmu.edu/~pstone/thesis

The page at this URL includes links to the CMUnited-97 simulator team source code and portions of the CMUnited-98 simulator team source code. Executables are also available.